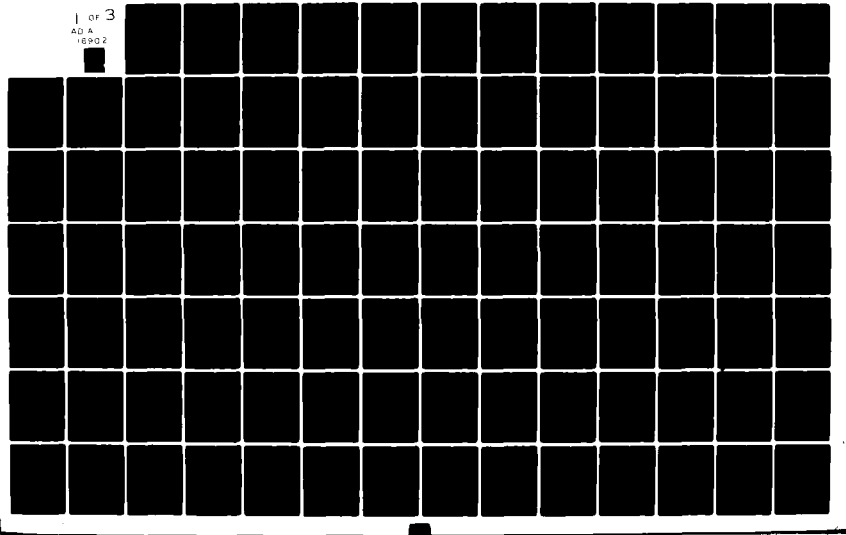
AD-A116 902 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2
REALTIME IMPLEMENTATION OF THE APC/SQ AND LPC-10 SPEECH CODING --ETC(U)
JUN 82 J J WOLF, K D FIELD, W H RUSSELL DCA100-80-C-0034
UNCLASSIFIED BBN-4855 NL



Bolt Beranek and Newman Inc.

(12)

Report No. 4855

Real-Time Implementation of the APC/SQ and LPC-10 Speech Coding Algorithms

Final Report

June 1982

Prepared for:
Defense Communications Agency

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BBN Report No. 4855	2. GOVT ACCESSION NO. AD-A116 902	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) REAL-TIME IMPLEMENTATION OF THE APC/SQ AND LPC-10 SPEECH CODING ALGORITHMS		5. TYPE OF REPORT & PERIOD COVERED Final Report August 1980 - June 1982
		6. PERFORMING ORG. REPORT NUMBER BBN Report No. 4855
7. AUTHOR(s) J.J. Wolf, K.D. Field, and W.H. Russell		8. CONTRACT OR GRANT NUMBER(s) DCA100-80-C-0034
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 10 Moulton Street Cambridge, MA 02238		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Agency Contract Management Division, Code 680 Washington, DC 20305		12. REPORT DATE June 1982 13. NUMBER OF PAGES 204
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Defense, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Speech coding, APC/SQ, LPC-10, 9.6 kb/s speech transmission, 2.4 kb/s speech transmission, medium-bandwidth speech transmission, narrow-bandwidth speech transmission, adaptive predictive coding, linear predictive coding, digital voice terminal, real-time speech coder, array processor

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the implementation of two standard speech compression algorithms, APC/SQ and LPC-10, as full-duplex real-time speech coders on the CSP Inc. MAP-300 array processing computer. The report documents in detail the MAP-300 and host computer software. It also includes an algorithmic description of APC/SQ.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Report No. 4855

REAL-TIME IMPLEMENTATION OF THE APC/SQ AND LPC-10 SPEECH CODING ALGORITHMS

J. J. Wolf, K. D. Field, W. H. Russell

June 1982

Prepared by:

Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, Massachusetts 02238

Prepared for:

Defense Communications Agency

TABLE OF CONTENTS

		Page
1.	INTRODUCTION	1
2.	REAL-TIME IMPLEMENTATIONS	3
2.1	APC/SQ AND LPC-10 OVERVIEW	3
2.1.1	System Function	3
2.1.2	System Components	3
2.1.3	System Design	4
2.2	APC/SQ SYSTEM IMPLEMENTATION	9
2.2.1	APC/SQ Transmitter	11
2.2.1.1	Begin Function List	14
2.2.1.2	Set G-flag	14
2.2.1.3	Monitor Peak Input Level	14
2.2.1.4	Analysis Frame History Updates	14
2.2.1.5	Remove DC Bias	15
2.2.1.6	TSRA/B is Empty	15
2.2.1.7	Preemphasis	15
2.2.1.8	Pitch Filter Analysis and Inverse Filter	16
2.2.1.9	Error Protection on Previous Frame Residual Data	16
2.2.1.10	Spectral Filter Analysis	17
2.2.1.11	Compute Frame and Sample Q's	18
2.2.1.12	Error Protection on Previous Frame Residual Data	19
2.2.1.13	Generate APC Residual	19
2.2.1.14	Error Protection on Previous Frame Parameter Data	20
2.2.1.15	Transmitter Data Collection	20

2.2.1.16 End of Function List 20
2.2.1.17 Error Protection and Bitstreaming 21
2.2.2 APC/SQ Receiver 23
2.2.2.1 Unbitstreaming and Error Correction 26
2.2.2.2 Begin Function List 27
2.2.2.3 Clear G-flag 27
2.2.2.4 Receiver Data Distribution 27
2.2.2.5 Synthesis Frame History Buffer Updates 28
2.2.2.6 Compute Sample Q's and Scale APC Residual 28
2.2.2.7 Spectral Synthesis 28
2.2.2.8 Decoding of Previous Frame Residual Data and Error Correction on Previous Frame Receiver Parameter Data 29
2.2.2.9 Pitch Synthesis 29
2.2.2.10 Deemphasis 30
2.2.2.11 Overflow Check 30
2.2.2.12 RSNKA/B is Full 30
2.2.2.13 End of Function List 31
2.3 LPC-10 SYSTEM IMPLEMENTATION 31
2.3.1 LPC-10 Transmitter 32
2.3.1.1 Begin Function List 34
2.3.1.2 Set G-flag 34
2.3.1.3 Analysis Frame History Updates 34
2.3.1.4 Long Term DC Removal and Zero Crossing Counts 35
2.3.1.5 Pitch and Voicing Analysis 35
2.3.1.6 Code Filter Parameters and Gather 38
2.3.1.7 Spectral Filter Analysis 38
2.3.1.8 End of Function List 39
2.3.1.9 Error Protection and Bitstreaming 39
2.3.2 LPC-10 Receiver 41
2.3.2.1 Unbitstreaming, Error Correction, Parameter Smoothing, and Decoding 44
2.3.2.2 Begin Function List 45
2.3.2.3 Clear G-flag 45
2.3.2.4 Synthesis Frame History Buffer Updates 46
2.3.2.5 Reflection to Predictor Coefficient Conversion 46
2.3.2.6 Epoch Generation 46
2.3.2.7 Pitch-Synchronous Synthesis 47
2.3.2.8 Unbitstreaming, Error Correction, and Decoding 47

2.3.2.9 End of Function List 47
2.4 SYSTEM HARDWARE 48
2.4.1 MAP-300 Hardware 48
2.4.2 Audio and Modem Interface Hardware 49
2.4.3 New 100 Hz High-Pass Filter 49
2.5 SYSTEM SOFTWARE 50
2.5.1 MAP-300 Software Components 52
2.5.1.1 CSPI-Supplied MAP-300 Software 54
2.5.1.2 BBN-Written MAP-300 Software 54
2.5.2 Host Computer Software Components 58
2.5.2.1 CSPI-Supplied Host Computer Software 59
2.5.2.2 BBN-written Host Computer Software 60
2.6 SYSTEM TIMING PERFORMANCE 66
2.6.1 APC/SQ System Timing 66
2.6.2 LPC-10 System Timing 68
2.7 IMPLEMENTATION CORRECTNESS ISSUES 71

3. SOFTWARE OPERATING PROCEDURES 83

3.1 SOFTWARE EXECUTION PROCEDURE 83
3.1.1 Optional Software Execution Procedure (No G-Flag) 86
3.1.2 Analog Input Level Setting 87
3.2 SOFTWARE INSTALLATION PROCEDURE 87

4. REFERENCES 91

APPENDIX A. ALGORITHMIC DESCRIPTION OF APC/SQ 93

APPENDIX B. BBN-WRITTEN MAP-300 FUNCTIONS FOR APC/SQ 121

APPENDIX C. BBN-WRITTEN MAP-300 FUNCTIONS FOR LPC-10 156

LIST OF FIGURES

FIG. 1.	SYSTEM COMPONENTS	6
FIG. 2.	SYSTEM STRUCTURE	7
FIG. 3.	BACKGROUND PROCESS LOOP	8
FIG. 4.	TRANSMITTER PROCESS	12
FIG. 5.	RECEIVER PROCESS	25
FIG. 6A.	ORIGINAL INPUT FILTERING CIRCUIT.	51
FIG. 6B.	HIGH-PASS FILTER AND SWITCH, INSERTED AT A-A' IN FIG. 6A	52
FIG. 7.	SKIRT RESPONSES OF TTE HIGH-PASS FILTER (SOLID LINE) AND THE HIGH-PASS FILTER DESCRIBED IN [9], (DASHED LINE).	53
FIG. 8.	MAP-300 MEMORY ORGANIZATION FOR APC/SQ	55
FIG. 9.	MAP-300 MEMORY ORGANIZATION FOR LPC-10	56
FIG. 10.	APC/SQ MAP-300 TIMING	67
FIG. 11.	LPC-10 MAP-300 TIMING	69
FIG. 12.	APC/SQ PARAMETER HISTOGRAM: FRAME Q	72
FIG. 13.	APC/SQ PARAMETER HISTOGRAM: SEGMENT Q'S	73
FIG. 14.	APC/SQ PARAMETER HISTOGRAM: RC1	74
FIG. 15.	APC/SQ PARAMETER HISTOGRAM: RC2	75
FIG. 16.	APC/SQ PARAMETER HISTOGRAM: RC3	76
FIG. 17.	APC/SQ PARAMETER HISTOGRAM: RC4	77
FIG. 18.	APC/SQ PARAMETER HISTOGRAM: PITCH TAP	78
FIG. 19.	APC/SQ PARAMETER HISTOGRAM: PITCH LAG	79

LIST OF TABLES

TABLE 1.	TSINK BUFFER FORMAT (APC/SQ)	22
TABLE 2.	TBITS BUFFER FORMAT (APC/SQ)	24
TABLE 3.	TSINK BUFFER FORMAT (LPC-10)	40
TABLE 4.	TBITS BUFFER FORMAT (LPC-10)	42

1. INTRODUCTION

The purpose of this project was the development of real-time implementations of two standard Government speech coding algorithms (APC/SQ and LPC-10) on a CSP Inc. (CSPI) MAP-300 signal processing computer. APC/SQ is an Adaptive Predictive Coder with Segmented Quantization that transmits coded speech at a data rate of 9.6 kb/s (kilobits/second). LPC-10 is a 10th order Linear Predictive Coder that transmits coded speech at a data rate of 2.4 kb/s. These implementations operate as real-time full duplex systems on the MAP-300 signal processing computer and associated hardware.

In the body of this report we present detailed documentation of the MAP-300 real-time implementations of the speech coding algorithms (Section 2) and instructions on the installation and use of the speech coder software (Section 3). Contained in appendices are an algorithmic description of APC/SQ (Version 09) (Appendix A), function descriptions of BBN-written MAP-300 modules for APC/SQ (Appendix B), and function descriptions of BBN-written MAP-300 modules for LPC-10 (Appendix C).

2. REAL-TIME IMPLEMENTATIONS

2.1 APC/SQ AND LPC-10 OVERVIEW

This section describes the function, components, and design of the two real-time speech coder implementations. Unless otherwise noted, the material in this section is common to both the APC/SQ and LPC-10 implementations. The later sections in this chapter describe in more detail the operation of each of the speech coder systems, the hardware and software used to construct them, and the real-time performance of the completed systems.

2.1.1 System Function

Each speech coder system is a full duplex terminal of a complete communication system. It is intended to be connected via a digital I/O link through a communication channel to an identical system. Each speech coder system functions simultaneously as both a transmitter and a receiver.

2.1.2 System Components

Each speech coder system contains both hardware and software elements. The hardware elements include a CSP Inc. MAP-300 array processor attached to a host computer, a handset including

microphone and earphone, a hookswitch, amplifiers, high-pass and low-pass filters, and digital line interfaces. The software elements include MAP-300 programs, which comprise the real-time software, and the program that runs on the host computer, which is used only to load and initialize the MAP software. Figure 1 is a block diagram of the system.

2.1.3 System Design

Each real-time speech coder consists of six separate foreground processes. The Transmitter requires an analog-in process, an analysis process, and a digital-out process. The Receiver requires a digital-in process, a synthesis process, and an analog-out process. Since all of these processes make use of the MAP-300 CSPU to some extent, a mechanism for scheduling them is necessary. Part of this mechanism is contained in the MAP hardware interrupt structures and the SNAP-II executive program. The rest is implemented using flags in conjunction with a background process running in the CSPU. Figure 2 is a diagram of these processes. The processes share data buffers and communicate the status of these buffers through flags. Since the four I/O processes operate continuously, double buffers are used to allow the necessary sharing.

Each I/O process includes an interrupt service routine,

which handles the flags and transfers data to or from the shared buffers. Two levels of double buffering are provided within each I/O process to maintain acceptable system performance under real-time exceptional conditions, for example, unusually long analysis or synthesis processing time, received data clock drift, received data interruption, or loss of received frame synchronization.

The background process is shown in Fig. 3. The ANALYZE and SYNTHESIZE modules are logically asynchronous; they respectively belong to the independent Transmitter and Receiver. However, the modules must be controlled by a sequential machine, the CSPU. The control strategy for allowing each module as much flexibility as possible results in a structure that executes a module only if that module's input and output buffers are available.

The Initialization module is the first module executed, and it is executed only once. It sets all buffer flags to indicate empty, loads and starts the programs in the various I/O Scroll processors, and enables interrupts from these processors.

Control then passes to the basic loop of the background process. This loop executes the ANALYZE and SYNTHESIZE modules when the required I/O buffers are available. For example, the ANALYZEA module will be executed only when the TSOURCEA buffer is full and the TBITSB buffer is empty.

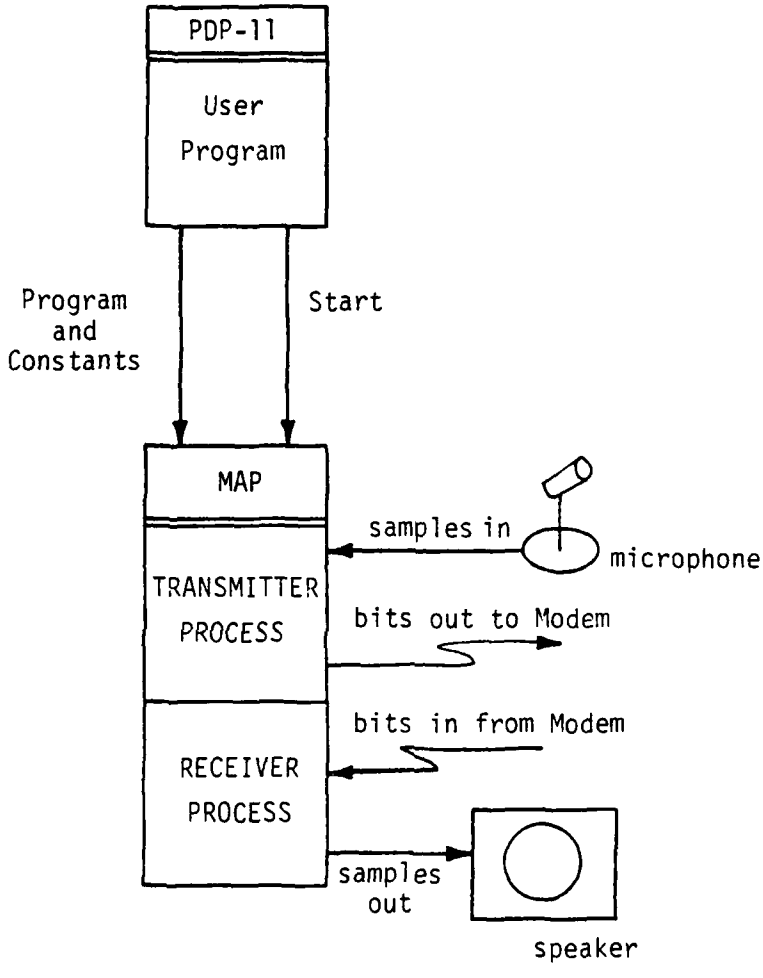


FIG. 1. SYSTEM COMPONENTS

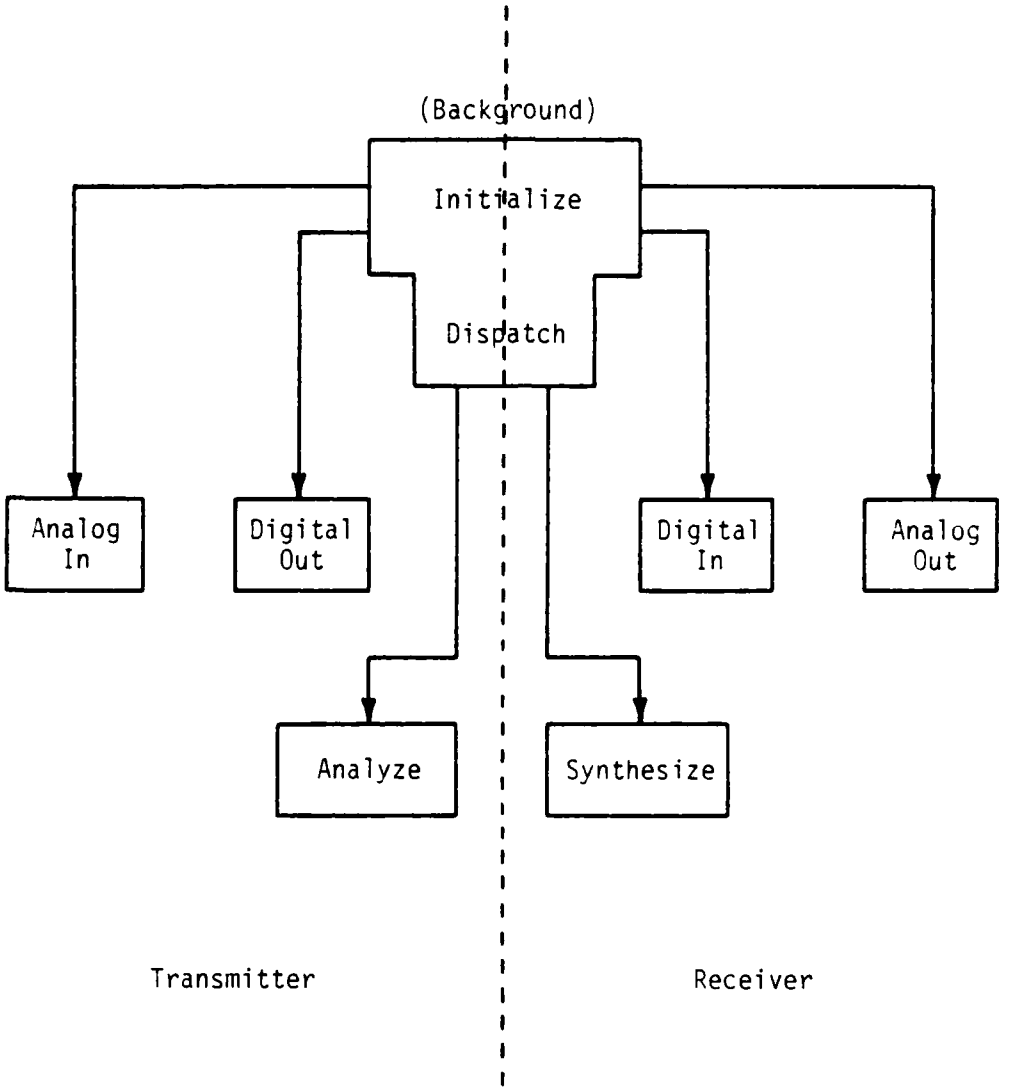


FIG. 2. SYSTEM STRUCTURE

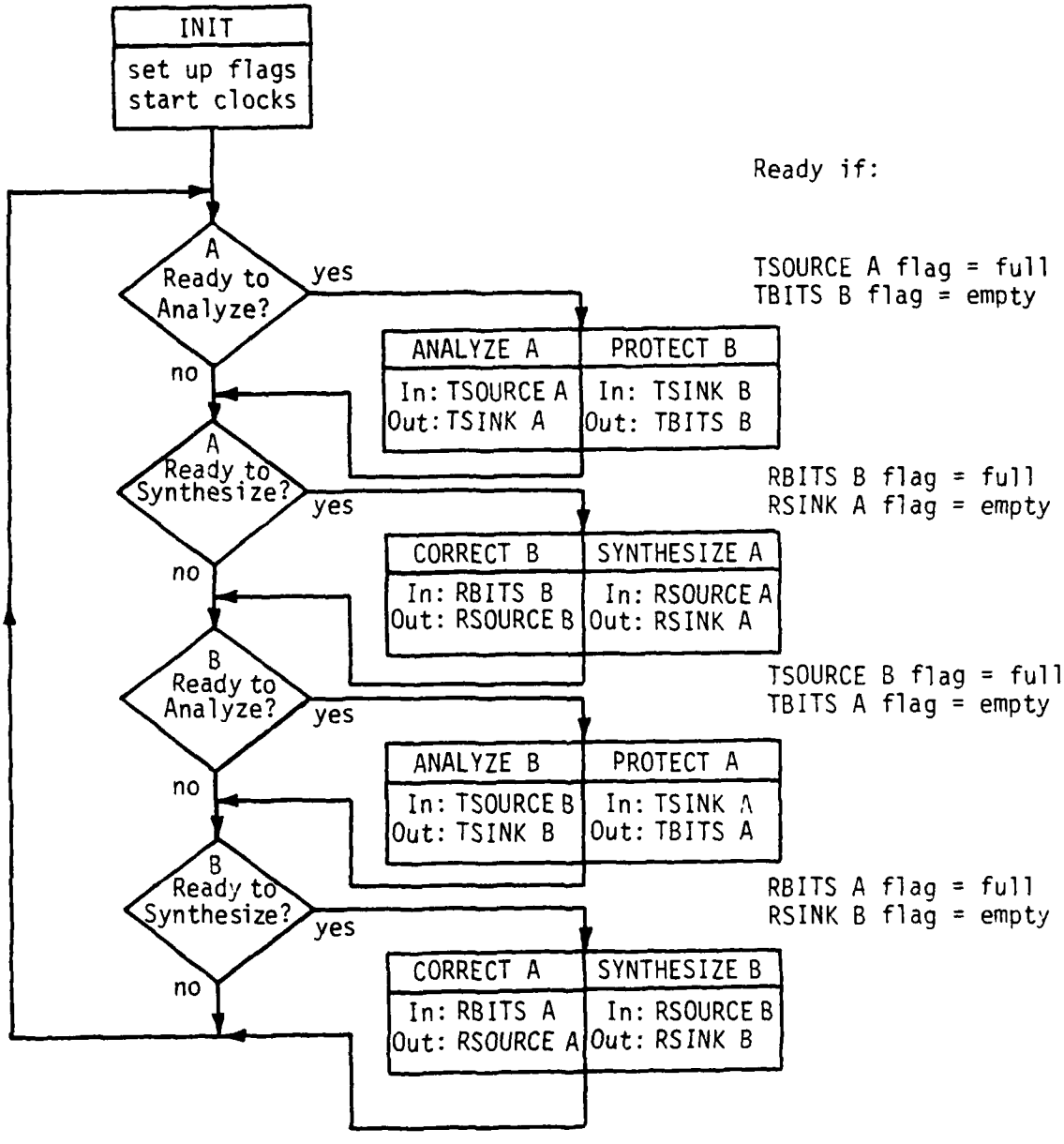


FIG. 3. BACKGROUND PROCESS LOOP

The ANALYZEA and ANALYZEB modules are functionally equivalent, differing only in their input and output buffers. The use of prebound functions for run-time efficiency precludes the run-time binding of input/output parameters of arithmetic functions. Therefore, two separate modules have been created to deal with the two sets of TSOURCE and TBITS buffers, A and B. Similarly, two separate SYNTHESIZE modules are required. (In the LPC-10 implementation, six separate SYNTHESIZE modules are required; see Section 2.3.2.)

All of the foreground processes are loosely coupled. Execution of neither the ANALYZE nor the SYNTHESIZE module is connected rigidly to any I/O process.

2.2 APC/SQ SYSTEM IMPLEMENTATION

The APC/SQ real-time MAP-300 speech coder system is a floating point implementation of the APC/SQ (Version 09) algorithm. This algorithm is defined by the Philco Ford Signal Processor (PFSP) fixed point implementation described in [9,10]. A Fortran bit-for-bit simulation of the PFSP implementation is described in [11]. An algorithmic description of the MAP-300 floating point APC/SQ implementation is given in Appendix A of this report.

There are several slight differences between the MAP-300 and PFSP implementations.

1. Due to hardware limitations, the MAP-300 implementation uses a speech signal sampling rate of 7680 Hz, or 192 samples per 25 ms frame (rather than the PFSP implementation's 7600 Hz, or 190 samples per frame). This modification affects APC/SQ's frame segmentation scheme, as well as its procedure for choosing 180 residual samples for transmission from the total frame of analysis samples.

    a. In the PFSP implementation, each frame is divided into ten segments of 19 samples each. Each segment is used to compute a segment energy from which a segment quantizer level is calculated. The MAP-300 implementation uses nine segments of 19 samples each and a tenth segment of 21 samples. The energy from this last segment is normalized by a factor of 19/21 prior to segment quantizer level computation.

    b. The PFSP implementation transmits only 180 of the 190 residual samples: every nineteenth residual sample is set to zero in the transmitter. These samples are not transmitted, but are reinserted as zeros by the receiver into the received frame of residual samples. The MAP-300 implementation transmits 180 residual samples of its 192 sample frame: every sixteenth sample is set to zero and not transmitted.

2. AGC sample scaling and predictor coefficient scaling has been removed in the MAP-300 implementation, since the floating point nature of this system implies automatic retention of maximal precision and essentially eliminates the possibility of arithmetic overflow.

3. The MAP-300 implementation uses a fractional speech sample range of -1.0 to +1.0, instead of the PFSP's integer range of -2048 to +2047. This affects constants throughout the algorithm, as well as the decoding table for frame quantizer level.

4. The single-tap pitch filter coefficient is maintained as an unscaled value in the MAP-300 implementation, instead of being scaled by 0.5.

5. The MAP-300 implementation transmits only a single sync bit per frame, allowing for transmission of all 20 reflection coefficient bits. The PFSP implementation transmits two sync bits per frame and only 19 reflection coefficient bits, sacrificing the low order bit of RC4.

The speech coder system functions as one terminal of a full duplex digital voice connection. It accepts voice input, digitizes it, and processes it. The processed speech is transmitted as a sequence of bits to a similar terminal. The system also accepts a sequence of bits representing speech transmitted from a remote terminal and processes this sequence to obtain synthetic voice output.

2.2.1 APC/SQ Transmitter

The Transmitter is shown in Fig. 4. The A/D input and coded bitstream output portions of the transmitter are virtually identical to the corresponding portions of two real-time speech coders previously developed by BBN, and they are described in the final reports documenting those coders [1, Vol. II; 8]. This section describes the analysis processing portion of the transmitter. Each analysis frame consists of 192 samples, corresponding to 25.0 ms of speech (at a sampling rate of 7.680 kHz). The ANALYZE Module implements the APC/SQ analysis

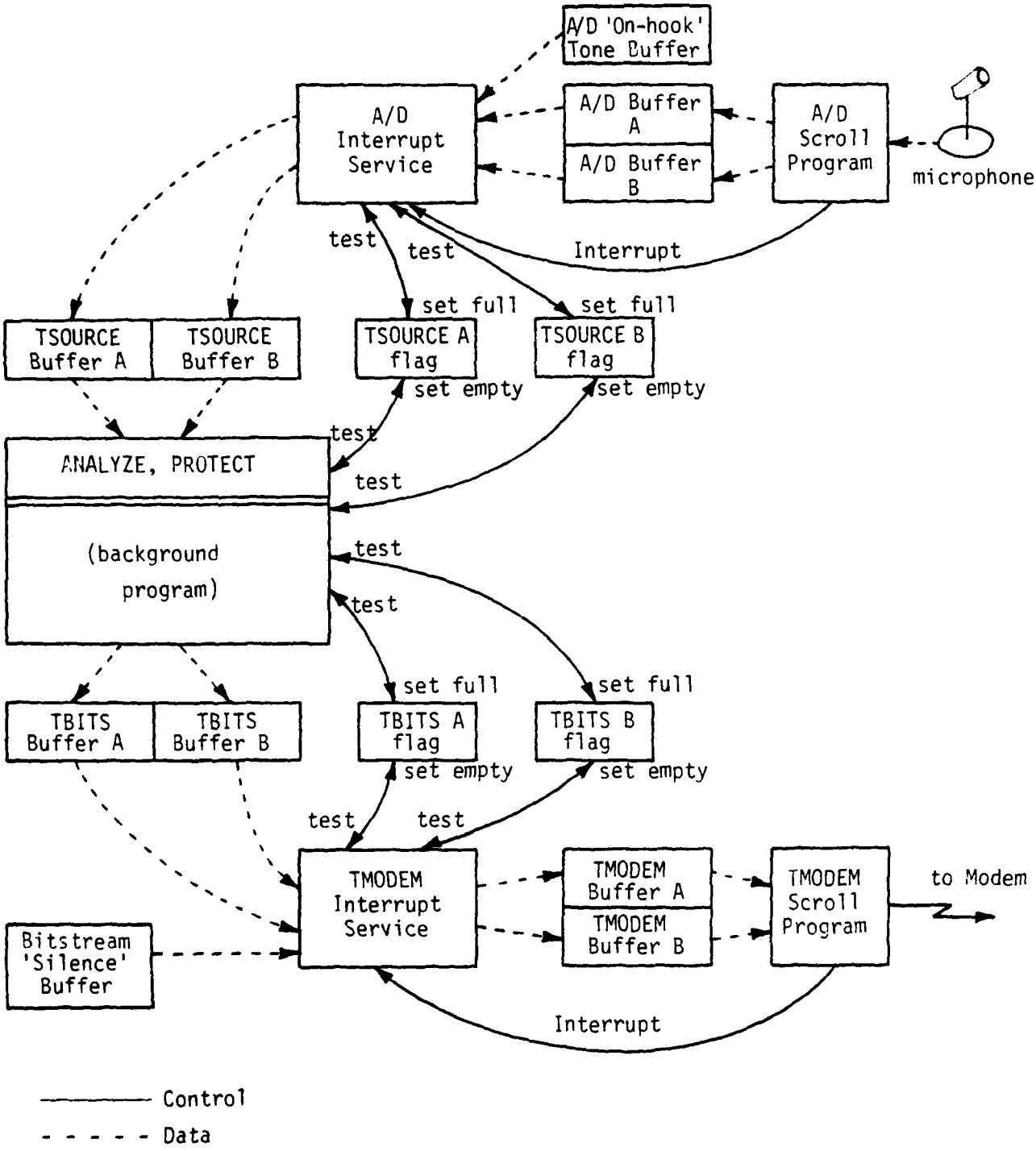


FIG. 4. TRANSMITTER PROCESS

algorithm described in Section 3 of Appendix A. As described in Section 2.1.3, there are two versions of the ANALYZE Module, one for each of the two sets of input/output buffers required for double buffering. Each version consists of a function list of MAP-300 function calls. The two versions (ANLZA and ANLZB) are identical in terms of the sequence of functions called and differ only in the parameters (buffers and/or scalars) passed to several of the functions.

This section describes, on a function by function basis, the ANLZA/ANLZB function lists, which appear in subroutine APCSQF of the speech coder host software. The specification of alternative parameters for the two function lists is indicated in the descriptions below by a '/' (e.g., 'TSRA/B' indicates TSRA for the ANLZA function list and TSRB for the ANLZB function list). Each function is either supplied by CSPI as part of the SNAP-II, Release 3.5 software system or written by BBN. CSPI-supplied functions are described in detail in CSPI documents [2,3,4]. BBN-written functions are described in detail in Appendix B of this report. The buffers and scalars passed as parameters to the functions are defined in subroutine APCSQC of the speech coder host software.

2.2.1.1 Begin Function List

MPBFL(ANLZA/B)

This CSPI function specifies the start of the indicated function list.

2.2.1.2 Set G-flag

MPGSC(IG3,ISET)

This BBN function causes general-purpose flag G3 to be set, indicating the start of ANLZ processing. This use of flag G3 is intended for system debugging, timing, and internal measurement and is not required for proper operation of the speech coding software.

2.2.1.3 Monitor Peak Input Level

ADPEAK(TSRA/B)

This BBN function serves to maintain a running maximum (in integer scalar TADPK) of the digitized speech input samples, allowing the user to properly adjust the analog input signal level for maximum A/D range without clipping. The maximum sample in the current input buffer is compared to the previous maximum, and TADPK is updated if necessary. TADPK is displayed (and reset to zero) during speech coder operation in response to a "T" command.

2.2.1.4 Analysis Frame History Updates

AHISTA(TSGQH,TSGQE,TER1H,TER1E,TPTCH,TPTC,TSPEH,TSPEE)

This BBN function performs frame history updates on four buffers. The last-frame data from the end of each

buffer is moved to the beginning of the buffer for use during current frame processing. (The buffers are referenced by TSGQH, TER1H, TPTCH, and TSPEH. The end portions of these buffers are referenced by TSGQE, TER1E, TPTC, and TSPEE.)

2.2.1.5 Remove DC Bias

REMDCA(TSPD,TDCSM,TSRA/B,TDCVL)

This BBN function removes from the input speech signal (TSRA/B) an estimate of the long-term DC bias, producing a DC-removed speech signal (TSPD). The input signal range is -1.0 to +1.0. The algorithm used for DC removal is specified in Section 3.1 of Appendix A.

2.2.1.6 TSRA/B is Empty

MPWT(PRCSR,AP)
MPIST(TSRFA/B,0)

These CSPI functions clear the appropriate buffer status flag (TSRFA or TSRFB) once the previous function has completed, indicating that the associated buffer can now be considered empty (i.e., ready to be filled with new input speech data).

2.2.1.7 Preemphasis

DFL22(TSPP,TPRC0,TSPD,TPRM0)

This CSPI function (Discrete 2-Pole 2-Zero Filter) preemphasizes the DC-removed speech (TSPD) (as indicated in Section 3.2 of Appendix A), with filter coefficients in four consecutive scalars starting at TPRC0, with filter history in four consecutive scalars starting at TPRM0, and with output preemphasized speech in TSPP.

2.2.1.8 Pitch Filter Analysis and Inverse Filter

The functions in this section perform pitch filter analysis and pitch inverse filtering on the preemphasized input speech, as indicated in Sections 3.3-3.5 of Appendix A.

PITCHA(TAMDF,TPITH,TSPT,TPITI,TAMIN,TAMAX)

This BBN function computes 60 AMDF values (TAMDF) from the extended frame of preemphasized input speech samples (TSPT), saves the minimum and maximum AMDF values in TAMIN and TAMAX, and computes and codes the pitch lag (in terms of number of speech samples) from the index of the minimum AMDF value, into TPITH (pitch lag) and TPITI (coded pitch lag).

TAPA(TSPP,TPITI,TPITH,TTAPI,TTAPH)

This BBN function computes a single-tap pitch predictor coefficient from the preemphasized input speech (TSPP), and generates a quantized pitch coefficient (TTAPH) and a coded pitch coefficient (TTAPI). The pitch lag (TPITH) computed in PITCHA is used in the pitch coefficient computation, and the coded pitch lag (TPITI) is modified for certain values of the coded pitch coefficient.

INVPFA(TER1,TPITH,TSPX,TTAPH)

This BBN function inverse pitch filters the preemphasized input speech (with last frame's 2nd APC-spin reconstructed speech as history) (TSPX), using the quantized pitch filter coefficient (TTAPH) and pitch lag (TPITH), and produces a first residual signal (TER1).

2.2.1.9 Error Protection on Previous Frame Residual Data

PRRES3(TBTB/A,TSNKB/A)

This BBN function bitstreams previous frame residual samples from the first 3/4 of TSNKB/A, depositing bitstreamed transmission data in TBTB/A. This function executes in the CSPU while function PITCHA is executing in the AP. (See Section 2.2.1.17 for a more detailed discussion of this module.)

2.2.1.10 Spectral Filter Analysis

The functions in this section perform spectral filter analysis on the first residual signal, as indicated in Section 3.6 of Appendix A.

SSMSQ(TR0,TER1,TRFSZ)

This CSPI function (Scalar Sum of Squares) is used to compute the zeroth autocorrelation divided by the frame size (TR0) of the first residual signal (TER1). TRFSZ is the reciprocal of the frame size.

COVMX(TCOVM,TER1,TCORV)

This function (written by CSPI and modified by BBN) computes the covariance matrix (TCOVM) and the correlation vector (TCORV) from the first residual signal (TER1).

CVMODA(TCOVM,TR0L,TCORV,TVPL,TCOVM,TCORV)

This BBN function modifies the covariance matrix (TCOVM) and correlation vector (TCORV) by adding to them scaled versions of a constant matrix and vector. The constant matrix and vector are scaled by the previous frame's zeroth autocorrelation term (TR0L) and the previous frame's spectrum loop energy reduction (TVPL).

MSFBS(TRC,TCORV,TCOVF,TCOVM)

This function (written by CSPI and modified by BBN) computes a set of pseudo-reflection coefficients (TRC)

from the modified covariance matrix (TCOVM) and correlation vector (TCORV). TCOVF is a work buffer.

RCCQA(TRCH,TRCI,TRC)

This BBN function codes and quantizes the reflection coefficients (TRC) into buffers TRCI and TRCH.

2.2.1.11 Compute Frame and Sample Q's

The functions in this section compute frame and sample Q's, as indicated in Sections 3.7-3.11 of Appendix A.

ESTQA(TQE,TRCH,TR0,TWK,TVPL)

This BBN function computes an estimate of the frame quantizer gain level (TQE) from the quantized reflection coefficients (TRCH), the zeroth autocorrelation term divided by frame size (TR0), and an empirical scale factor (TWK). The current frame spectrum loop energy reduction (TVPL) is computed for use next frame in function CVMODA.

PCOEF(TPC,TRCH)

This BBN function converts reflection coefficients (TRCH) into predictor coefficients (TPC).

APC1A(TPC,TSPP,TPTCH,TSPEH,TSGSM,TTAPH,TFSUM,TSGMX)

This BBN function performs the first spin of the APC loop, resulting in segment sums (TSGSM), frame sum (TFSUM), and the maximum segment sum (TSGMX). Predictor coefficients (TPC), the pitch filter coefficient (TTAPH), the pitch lag (TPITH), the frame Q estimate (TQE), and the preemphasized speech signal (TSPP) are used as input. (TPITH and TQE are input implicitly via their memory locations with respect to TTAPH.) TPTCH and TSPEH contain pitch filter and spectral filter histories.

CALQA(TSGQ,TQRI,TSGNI,TQE,TSGSM,TFSUM,TSGMX)

This BBN function computes a refined frame quantizer gain level, codes it (TQRI), and uses it to compute segment quantizer levels (TSGQ) and coded segment normalization factors (TSGNI). The frame Q estimate (TQE), segment sums (TSGSM), frame sums (TFSUM), and maximum segment sum (TSGMX) are used as input.

SMPQA(TSAMQ,TSGQH)

This BBN function computes sample-by-sample quantizer levels (TSAMQ) from segment quantizer levels (including the last segment Q from the previous frame) (TSGQH).

2.2.1.12 Error Protection on Previous Frame Residual Data

PRRES1(TBTB/A,TSNKB/A)

This BBN function bitstreams previous frame residual samples from the final 1/4 of TSNKB/A, depositing bitstreamed transmission data in TBTB/A. This function executes in the CSPU while function APC1A is executing in the AP. (See Section 2.2.1.17 for a more detailed discussion of this module.)

2.2.1.13 Generate APC Residual

APC2A(TPC,TSPP,TPTCH,TSPEH,TSAMQ,TSPR,TR2IA/B,TTAPH)

This BBN function performs the second spin of the APC loop (as indicated in Section 3.12 of Appendix A), resulting in the coded APC residual signal (TR2IA/B, which is equivalent to the beginning of TSNKA/B). Predictor coefficients (TPC), sample Q's (TSAMQ), the pitch filter coefficient (TTAPH), the pitch lag (TPITH), and the preemphasized speech signal (TSPP) are used as input. (TPITH is input implicitly via its memory location with respect to TTAPH.) TPTCH and TSPEH contain

pitch filter and spectral filter histories. TSPR is output as the reconstructed 2nd APC-spin speech.

2.2.1.14 Error Protection on Previous Frame Parameter Data

```
PROPAR(TBTB/A,TSNKB/A)
MPIST(TBTFB/A,1)
```

These functions (PROPAR is a BBN function; MPIST is a CSPI function) perform error-protection and bitstreaming on the previous frame's parameter data (TSNKB/A), depositing bitstreamed transmission data in TBTB/A. The appropriate buffer status flag (TBTFB or TBTFA) is set, indicating that the associated buffer is now full (i.e., ready to be transmitted). These functions execute in the CSPU while function APC2A is executing in the AP. (See Section 2.2.1.17 for a more detailed discussion of this module.)

2.2.1.15 Transmitter Data Collection

```
GTHRA(TSNKA/B,TQRI,TSGNI,
    TTAPI,TRCI,TPITI)
```

This BBN function collects all coded parameter transmission data into a single buffer (TSNKA/B). (The coded residual data has been placed directly into TSNKA/B by the APC2A function.)

2.2.1.16 End of Function List

```
MPEFL(ANLZA/B)
```

This CSPI function specifies the end of the indicated function list.

2.2.1.17 Error Protection and Bitstreaming

The PROPAR, PRRES3, and PRRES1 modules take as input a TSNK buffer (either TSNKA or TSNKB) containing quantized and coded analysis parameters and residual samples and produce as output a TBITS buffer (again, TBTA or TBTB according to the TSINK buffer designation) in which:

1. certain high-order data bits have been grouped together and protected with a (21,16) modified Hamming code that corrects a single-bit error and detects a double-bit error;
2. the data to be transmitted has been "bitstreamed", one bit per half-word, in the form used by the TMODEM scroll program;
3. histogram information of the coded analysis parameters has been recorded.

These operations are performed in the CSPU, and therefore they can be executed concurrently with an AP operation. These operations are implemented as three separate CSPU modules so that each module's execution time is hidden by that of an AP module. The PROPAR module performs the error-protection and bitstreaming of the analysis parameters (gain, segment-Q's, pitch (tau), pitch filter coefficient (alpha), and four reflection coefficients), and it also records the histogram information for those parameters. PRRES3 bitstreams the first 3/4 of the residual samples, and PRRES1 bitstreams the last 1/4.

The format of the TSINK buffer is shown in Table 1, along with the number of bits per parameter and the number of high-order bits protected by the (21,16) code. (SQC0-SQC9 denote the ten segmented-quantization levels, Q the overall gain, alpha the pitch filter coefficient, and K1-K4 the reflection coefficients.)

Word	Parameter	No. of bits	Bits protected
0-191	Residual samples	1	0
192-201	SQC0-SQC9	2	0
202	Q	5	2
203	Alpha	3	1
204	K1	5	1
205-207	K2-K4	5	2
208	Pitch	6	6

TABLE 1. TSINK BUFFER FORMAT (APC/SQ)

The TBITS buffer contains one data bit in the rightmost bit of each 16-bit half-word. The format of the TBITS buffer (also the frame of data transmitted) is shown in Table 2. (Bits of coded parameters are numbered starting with bit 0 on the right. An asterisk denotes that the next-higher residual sample is omitted from the transmitted bitstream.)

A histogram-gathering function was included in the PROPAR module for gathering statistics on the effectiveness of the quantization tables for the analysis parameters and for verifying correct transmitter operation. A simple Fortran program

(HISTAPC) was implemented to read the histogram buffers defined on Bus 1 and list them in a text file.

2.2.2 APC/SQ Receiver

The Receiver is similar in structure to the Transmitter. The Receiver is shown in Fig. 5.

The coded bitstream input, frame synchronization, and D/A output portions of the Receiver are virtually identical to the corresponding portions of two real-time speech coders previously developed by BBN, and they are described in the final reports documenting those coders [1, Vol. II; 8]. This section describes the synthesis processing portion of the Receiver.

The SYNTHESIZE Module implements the APC/SQ synthesis algorithm described in Section 4 of Appendix A. As described in Section 2.1.3, two versions of the SYNTHESIZE Module exist, one for each of the two sets of input/output buffers required for double buffering. Each version consists of a function list of MAP-300 function calls. The two versions (SYNZA and SYNZB) are identical in terms of the sequence of functions called and differ only in the parameters (buffers and/or scalars) passed to several of the functions.

As in the case of Section 2.2.1, this section describes, on

BIT	DATUM	BIT	DATUM	BIT	DATUM	BIT	DATUM	BIT	DATUM
0	E8	50	E83	100	E158*	150	E51	200	E126*
1	E7	51	RC3-2	101	E157	151	CKT-3	201	E125
2	E6	52	E82	102	E156	152	E50	202	E124
3	RC1-0	53	E81	103	TAU-3	153	E49	203	SQC5-1
4	E5	54	E80	104	E155	154	E48	204	E152
5	E4	55	RC3-3	105	E154	155	CKT-4	205	E151
6	E3	56	E78*	106	E153	156	E75	206	E150
7	RC1-1	57	E77	107	TAU-4	157	E74	207	SQC6-0
8	E2	58	E76	108	E181	158	E73	208	E149
9	E1	59	RC3-4	109	E180	159	SQC0-0	209	E148
10	E0	60	E104	110	E179	160	E72	210	E147
11	RC1-2	61	E103	111	TAU-5	161	E71	211	SQC6-1
12	E27	62	E102	112	E178	162	E70	212	E146
13	E26	63	RC4-0	113	E177	163	SQC0-1	213	E145
14	E25	64	E101	114	E176	164	E69	214	E144
15	RC1-3	65	E100	115	Q-0	165	E68	215	SQC7-0
16	E24	66	E99	116	E174*	166	E67	216	E171
17	E23	67	RC4-1	117	E173	167	SQC1-0	217	E170
18	E22	68	E98	118	E172	168	E94*	218	E169
19	RC1-4	69	E97	119	RC4-4	169	E93	219	SQC7-1
20	E21	70	E96	120	E18	170	E92	220	E168
21	E20	71	RC4-2	121	E17	171	SQC1-1	221	E167
22	E19	72	E123	122	E16	172	E91	222	E166
23	RC2-0	73	E122	123	Q-1	173	E90	223	SQC8-0
24	E46*	74	E121	124	E14*	174	E89	224	E165
25	E45	75	RC4-3	125	E13	175	SQC2-0	225	E164
26	E44	76	E120	126	E12	176	E88	226	E163
27	RC2-1	77	E119	127	Q-2	177	E87	227	SQC8-1
28	E43	78	E118	128	E11	178	E86	228	E190*
29	E42	79	ALPHA-0	129	E10	179	SQC2-1	229	E189
30	E41	80	E117	130	E9	180	E114	230	E188
31	RC2-2	81	E116	131	Q-3	181	E113	231	SQC9-0
32	E40	82	E115	132	E37	182	E112	232	E187
33	E39	83	ALPHA-1	133	E36	183	SQC3-0	233	E186
34	E38	84	E142*	134	E35	184	E110*	234	E185
35	RC2-3	85	E141	135	Q-4	185	E109	235	SQC9-1
36	E66	86	E140	136	E34	186	E108	236	E184
37	E65	87	ALPHA-2	137	E33	187	SQC3-1	237	E183
38	E64	88	E139	138	E32	188	E107	238	E182
39	RC2-4	89	E138	139	CKT-0	189	E106	239	SYNC
40	E62*	90	E137	140	E30	190	E105		
41	E61	91	TAU-0	141	E29	191	SQC4-0		
42	E60	92	E136	142	E28	192	E133		
43	RC3-0	93	E135	143	CKT-1	193	E132		
44	E59	94	E134	144	E56	194	E131		
45	E58	95	TAU-1	145	E55	195	SQC4-1		
46	E57	96	E162	146	E54	196	E130		
47	RC3-1	97	E161	147	CKT-2	197	E129		
48	E85	98	E160	148	E53	198	E128		
49	E84	99	TAU-2	149	E52	199	SQC5-0		

TABLE 2. TBITS BUFFER FORMAT (APC/SQ)

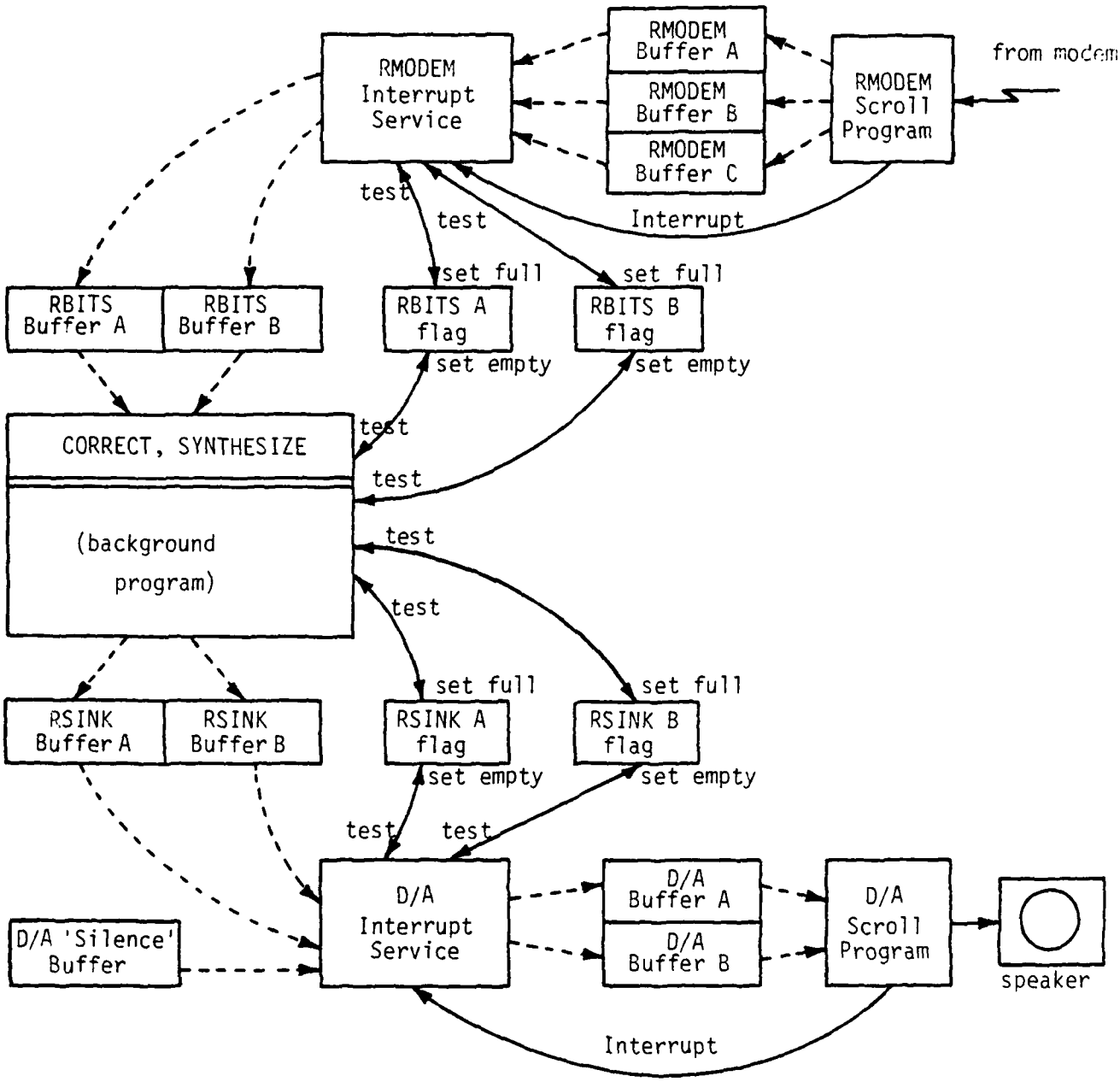


FIG. 9. RECEIVER PROCESS

a function by function basis, the SYNZA/SYNZB function lists, which also appear in subroutine APCSQF.

2.2.2.1 Unbitstreaming and Error Correction

The CORPAR and DECRES modules take as input an RBITS buffer containing a frame of bitstream data input from the modem and produce as output an RSOURCE buffer containing error-corrected and decoded floating point values of synthesis parameters and residual samples, ready for immediate use by the speech coder synthesizer.

The format of the RBITS buffer is the same as the TBITS buffer, shown in Table 2 above.

The data bits are regrouped into coded values for each parameter. New check bits (for the (21,16) code) are computed from the received parameters and compared with the received check bits. If one error is detected, it is corrected; if two errors are detected, then the protected parameters (Q, alpha, K1-K4 only, not pitch or SQC0-SQC9) from the previous frame are substituted for the current frame's parameters. The coded values are then decoded by lookup in parameter-specific decoding tables of floating-point values. The format of the RSOURCE buffer is similar to that of the TSINK buffer (Table 1), except that the half-word coded TSINK values are replaced by full-word (32-bit) floating point values.

As in the case of the error-protection and bitstreaming operation, the unbitstreaming, error-correction, and decoding operation is performed by two separate modules, which are part of the SYNTHESIZE function list. CORPAR does the operations on the analysis parameters, while DECRES performs unbitstreaming and decoding on the residual samples. Unlike the other "bit-pushing" modules, DECRES is an AP module.

2.2.2.2 Begin Function List

MPBFL(SYNZA/B)

This CSPI function specifies the start of the *indicated function list.*

2.2.2.3 Clear G-flag

MPGSC(IG3,ICLR)

This BBN function causes general purpose flag G3 to be cleared, indicating the start of SYNZ processing. This use of flag G3 is intended for system debugging, timing, and internal measurement, and is not required for proper operation of the speech coding software.

2.2.2.4 Receiver Data Distribution

DEALA(RSRA/B,RQR,RSGN,RTAP,RRC,RPIT)

This BBN function distributes all decoded parameter data from a single buffer (RSRA/B) to separate buffers and scalars. (The decoded residual data is read directly from RSRA/B by the VMUL function.)

2.2.2.5 Synthesis Frame History Buffer Updates

SHISTA(RSGQH,RSGQE,RPTCH,RPTC,RER1H,RER1E)

This BBN function performs frame history updates on three buffers. The last-frame data from the end of each buffer is moved to the beginning of the buffer for use during current frame processing. (The buffers are referenced by RSGQH, RPTCH, and RER1H. The end portions of these buffers are referenced by RSGQE, RPTC, and RER1E.)

2.2.2.6 Compute Sample Q's and Scale APC Residual

The functions in this section scale the decoded APC residual by sample-by-sample quantizer levels, as indicated in Sections 4.2 and 4.3 of Appendix A.

RCALQA(RSAMQ,RQR,RSGN,RSGQH)

This BBN function computes sample-by-sample quantizer levels (RSAMQ) from the decoded frame quantizer level (RQR) and the decoded segment normalization levels (RSGN). RSGQH contains the segment Q's, including the last segment Q from the previous frame, which are both written and read by this module.

VMUL(RR2Q,1,RR2A/B,0,RSAMQ,0)

This CSPI function scales the decoded APC residual (RR2A/B) by the sample-by-sample quantizer level (RSAMQ), producing a scaled APC residual (RR2Q).

2.2.2.7 Spectral Synthesis

The functions in this section perform spectral synthesis as indicated in Sections 4.4 and 4.5 of Appendix A.

PCOEF(RPC,RRC)

This BBN function converts reflection coefficients (RRC) into predictor coefficients (RPC).

DFL0N(RER1,RR2Q,RPC)

This BBN function executes a 10-pole spectral synthesis filter on the scaled APC residual (RR2Q), using the predictor coefficients (RPC) as filter coefficients, and puts the synthesized first residual signal in RER1.

2.2.2.8 Decoding of Previous Frame Residual Data and Error Correction on Previous Frame Receiver Parameter Data

DECRES(RR2B/A,RBTB/A)
CORPAR(RSRB/A,RBTB/A)
MPWT(PRCSR,AP)
MPIST(RBTFB/A,0)

These functions (DECRES and CORPAR are BBN functions; MPWT and MPIST are CSPI functions) decode the previous frame received residual samples and error-correct the previous frame received parameter data (from RBTB/A), depositing decoded data in RSRB/A (RR2B/A is equivalent to the beginning of RSRB/A). The appropriate buffer status flag (RBTFB or RBTFA) is cleared, indicating that the associated buffer can now be considered empty (i.e., ready to be filled with new received coded data). The last three functions execute in the CSPU while DFL0N and DECRES execute in the AP. (See Section 2.2.2.1 for a more detailed discussion of these modules.)

2.2.2.9 Pitch Synthesis

PTSYNA(RPTC,RPIT,RER1,RTAP)

This BBN function executes a single-tap pitch synthesis filter (as indicated in Section 4.5 of Appendix

A), using the decoded pitch lag (RPIT), the decoded pitch tap (RTAP), and the synthesized first residual (RER1), and producing a synthesized preemphasized speech signal (RPTC).

2.2.2.10 Deemphasis

DFL22(RSPD,RDEC0,RPTC,RDEM0)

This CSPI function (Discrete 2-Pole 2-Zero Filter) deemphasizes the synthesized speech signal (RPTC) (as indicated in Section 4.6 of Appendix A), with filter coefficients in four consecutive scalars starting at RDEC0, with filter history in four consecutive scalars starting at RDEM0, and with output deemphasized synthesized speech in RSPD.

2.2.2.11 Overflow Check

OVFLWA(RSNKA/B,ROVPT,RSPD)

This BBN function checks each deemphasized synthesized speech sample (RSPD) for overflow (as indicated in Section 4.7 of Appendix A), and replaces any overflow sample by the previous output sample. The output synthesized speech is in RSNKA/B. ROVPT is the last output sample from the previous frame, in case the first sample of the current frame overflows.

2.2.2.12 RSNKA/B is Full

MPWT(PRCSR,AP)
MPIST(RSNFA/B,1)

These CSPI functions set the appropriate buffer status flag (RSNFA or RSNFB) once the previous function has completed, indicating that the associated buffer is full (i.e., ready to be emptied).

2.2.2.13 End of Function List

MPEFL(SYNZA/B)

This CSPI function specifies the end of the indicated function list.

2.3 LPC-10 SYSTEM IMPLEMENTATION

The LPC-10 real-time MAP-300 speech coder system is a floating point implementation of the LPC-10 (Version 44) algorithm. This algorithm is defined by the Philco Ford Signal Processor (PFSP) fixed point implementation described in [9,12].

There are several slight differences between the MAP-300 and PFSP implementations.

1. AGC sample scaling and predictor coefficient scaling has been removed in the MAP-300 implementation, since the floating point nature of this system implies automatic retention of maximal precision and essentially eliminates the possibility of arithmetic overflow.

2. The MAP-300 implementation uses a fractional speech sample range of -1.0 to +1.0, instead of the PFSP's integer range of -2048 to +2047. This affects constants throughout the algorithm.

3. In the synthesis algorithm, if the previous pitch period was shorter than the 40-sample table of values used as voiced excitation for the synthesis filter, the remaining "tail" is scaled by a factor and added to the voiced excitation of the current pitch period; the scale factor used in this operation is the ratio of the previous period's RMS value to the current period's RMS

value. The overlap-and-add of the excitation signal is documented in [12], but the scaling of the "tail" values is not described there at all; it was discovered only through a detailed examination of the PFSP code, well after the implementation of the LPC-10 receiver had been fixed and mostly completed. Although the "tail scaling" appears to be a simple modification to the excitation generation, it could not be implemented without a major change to the MAP-300 implementation and without adding to the execution time, so this was not done. The omission of the "tail scaling" affects the synthesized output only during voiced speech for fundamental frequencies in the range 250-400 Hz, and even then, we feel that its effect on the output speech is minor.

The speech coder system functions as one terminal of a full duplex digital voice connection. It accepts voice input, digitizes it, and processes it. The processed speech is transmitted as a sequence of bits to a similar terminal. The system also accepts a sequence of bits representing speech transmitted from a remote terminal and processes this sequence to obtain synthetic voice output.

2.3.1 LPC-10 Transmitter

The LPC-10 Transmitter is identical in organization to the APC/SQ Transmitter, which is illustrated in Fig. 4. The A/D input and coded bitstream output portions of the transmitter are virtually identical to the corresponding portions of two real-time speech coders previously developed by BBN, and they are described in the final reports documenting those coders [1, Vol.

II; 8]. This section describes the analysis processing portion of the transmitter. Each analysis frame of input speech consists of 180 samples, corresponding to 22.5 ms (at a sampling rate of 8.0 kHz). The ANALYZE Module implements the LPC-10 analysis algorithm described in [9,12]. As described in Section 2.1.3, there are two versions of the ANALYZE module, one for each of the two sets of input/output buffers required for double buffering. Each version consists of a function list of MAP-300 function calls. The two versions (ANLZA and ANLZB) are identical in terms of the sequence of functions called and differ only in the parameters (buffers and/or scalars) passed to several of the functions.

This section describes, on a function by function basis, the ANLZA/ANLZB function lists, which appear in subroutine LPC10F of the speech coder host software. The specification of alternative parameters for the two function lists is indicated in the descriptions below by a '/' (e.g., 'TSRA/B' indicates TSRA for the ANLZA function list and TSRB for the ANLZB function list). Each function is either supplied by CSPI as part of the SNAP-II Release 3.5 software system or written by BBN. CSPI-supplied functions are described in detail in CSPI documents [2,3,4]. BBN-written functions are described in detail in Appendix C of this report. The buffers and scalars passed as parameters to the functions are defined in subroutine LPC10C of the speech coder host software.

2.3.1.1 Begin Function List

MPBFL(ANLZA/B)

This CSPI function specifies the start of the indicated function list.

2.3.1.2 Set G-flag

MPGSC(IG3,ISET)

This BBN function causes general-purpose flag G3 to be set, indicating the start of ANLZ processing. This use of flag G3 is intended for system debugging, timing, and internal measurement and is not required for proper operation of the speech coding software.

2.3.1.3 Analysis Frame History Updates

SHISTA(TSDCH,TSDCE,TSLPH,TSLPE,TSIVH,TSIVE)

This BBN function performs frame history updates on three buffers. The last-frame data from the end of each buffer is moved to the beginning of the buffer for use during current frame processing. (The buffers are referenced by TSDCH, TSLPH, and TSIVH. The end portions of these buffers are referenced by TSDCE, TSLPE, and TSIVE.)

AHISTA(TPT3,TPT2,TPT2,TPT1,TRMS,TMSE,
TRC,TRCE)

This BBN function performs frame history update on four buffers: TPT3, TPT2, TRMS, and TRC. Note that the data in buffer TPT2 is moved into buffer TPT3 and then the data in TPT1 replaces the data in buffer TPT2. History updates for buffers TRMS and TRC are as in SHISTA (Buffers are TRMS and TRC, and the end portions of these buffers are TRMSE and TRCE respectively).

2.3.1.4 Long Term DC Removal and Zero Crossing Counts

DCRZCL(TSDCH,TDCS,TSRA/B,TDCV,TNZ1,TDTH)

This BBN function removes the long term dc bias from the input speech (TSRA/B). TDCS and TDCV are scalars holding the running dc sum and running dc value. The dc-removed speech is output to buffer TSDCH. This function also computes from buffer TSDCH the first half-frame zero crossing count (TNZ1), second half-frame zero crossing count (TNZ2), and their sum (total frame zero crossing count (TNZT)). TNZ1, TNZ2, and TNZT are three contiguous scalars. TDTH is the level of a square wave signal added to the input (TSDCH) to force zero crossings for low-level input.

DCRZCL also serves to maintain a running maximum (in integer scalar TADPK) of the digitized speech input samples, allowing the user to adjust properly the analog input signal level for maximum A/D range without clipping. The maximum sample in the current input buffer is compared to the previous maximum, and TADPK is updated if necessary. TADPK is displayed (and reset to zero) during speech coder operation in response to a "T" command.

2.3.1.5 Pitch and Voicing Analysis

The functions in this section perform signal processing and decision logic for the estimation of voicing for the current frame and for pitch and voicing for the frame two in the past, which is about to be transmitted.

LPBUTL(TSLS,TSDC,TCLP,TSLP)

This BBN function low pass filters the input TSDC (long term dc removed signal). The filter is a 4th order Butterworth recursive filter with taps defined in TCLP. The filtered signal TSLS is then scaled by a constant factor (= 1/16), and the filtered, scaled signal is output into buffer TSLP.

INVFTL(TSIV,TIVP,TSLEN,TSLOD)

This BBN function performs an adaptive 2nd order inverse filtering operation on the low pass filtered speech in buffers TSLEN and TSLOD (TSLEN and TSLOD contain the even and odd low-pass filtered samples respectively). The parameters of the inverse filter are computed by this function and stored in buffer TIVP. Three autocorrelation terms are stored in TIVP(0-2), two reflection coefficients are stored in TIVP(3-4), and two predictor coefficients are stored in TIVP(5-6). TSIV contains the inverse filtered signal.

MPIST(TSRFA/B,0)

This CSPI function clears the appropriate buffer status flag (TSRFA or TSRFB), indicating that the associated buffer can now be considered empty (i.e., ready to be filled with new input speech data). The MPIST is done at this point to ensure that it happens after the completion of DCRZCL.

SSMABL(TSLP1,TEM1,TSLP2,TEM2)

This BBN function computes the sum of the absolute values of the first half-frame samples in TSLP1, then scales this sum by 2**11 (to adjust for fractional samples ranging from -1 to +1, instead of integer samples ranging from -2048 to 2047). The scaled sum is converted to integer format and output into the left half-word of scalar TEM1. It then does a similar operation on the second half-frame samples in TSLP2, with output into TEM2.

PITCHA(TAMF,TPA,TSIVH,TPB,TAMN,TAMX)

This BBN function computes 60 AMDF values (TAMF) from the extended frame of preemphasized input speech samples (TSIVH) and saves the minimum and maximum AMDF values in scalars TAMN and TAMX. (PITCHA also computes and codes the pitch lag from the index of the minimum AMDF value into TPA and TPB, but these values are not used in LPC-10).

EPROL(TBTB/A,TSNKB/A)
MPIST(TBTFB/A,1)

These functions (EPROL is a BBN function; MPIST is a CSPI function) perform error-protection and bitstreaming on the previous frame's coded parameter data (TSNKB or TSNKA), depositing bitstreamed transmission data in TBTB/A. The appropriate buffer status flag (TBTFB or TBTFA) is set, indicating that the associated buffer is now full (i.e., ready to be transmitted). These functions execute in the CSPU while function PITCHA is executing in the AP. (See Section 2.3.1.9 for a more detailed discussion of this module.)

MPWT(PRCSR,AP)
VNEWL(TNZ1,TAMN,TDTH,TVD1)

This BBN function (VNEWL) computes voicing decisions for the (two halves of the) current frame (saved internally), and refines them for the frame two in the past, which is the one about to be transmitted. The CSPI function MPWT insures that VNEWL (a CSPU function) does not execute until PITCHA (an AP function) has completed and has generated its outputs TAMN and TAMX. TNZ1 is the first of five consecutive input scalars (TNZ1, TNZ2, TNZT, TEM1, TEM2) containing (in integer format) zero crossing counts for two half frames and for the whole frame and scaled energy measures for two half frames. TAMN is the first of two consecutive input real scalars (TAMN, TAMX), containing the current frame minimum and maximum AMDF values. TDTH is an output scalar (in integer format) containing the "dither factor", which is used by DCRZCL. TVD1 is the first of three consecutive real scalars, containing voicing decisions for two half-frames of the frame to be transmitted and for the first half of the current frame.

DYNPRL(TWFN,TALX,TAMF,TVD1,TPT1,TAMN,TWMN)

This BBN function computes the pitch for both the current and the transmitted frames, using a dynamic programming algorithm. TWFN is the "winner" function buffer. TALX is the scalar confidence factor "alpha", followed by an output scalar containing a scaled "alpha". TAMF is the input AMDF function buffer. TVD1 is the

first of three consecutive input scalars, containing the voicing states for the two transmitted half-frames and for the first half of the current frame. TPTl is the first of three consecutive (fixed length) buffers, containing current, past, and transmitted frame data. TAMN is the minimum value of the AMDF buffer. TWMN is the first of six consecutive output scalars, containing the minimum and maximum of the updated "winner" function, the pitch code, the transmitted frame pitch index, the current frame pitch index, and the current frame pitch lag (in samples).

2.3.1.6 Code Filter Parameters and Gather

RCCGTL(TSNKA/B,TPTC,TRC,TRMS)

This BBN function codes the reflection coefficients (TRC) computed by the MSFBS function two frames earlier, and gathers all coded parameters, including pitch (TPTC) and gain (TRMS), into a single output buffer (TSNKA/B).

2.3.1.7 Spectral Filter Analysis

The functions in this section perform spectral filter analysis on the dc-removed signal of the current frame.

APHRML(TSAN,TNFR,TSDCH,TFIN,TRMS,TVDS,TPLC)

This BBN function computes the location of a quasi-pitch-synchronous inverval for LPC analysis. For voiced frames (voicing decision TVDS=2), the interval is determined from the location (in samples) of the last sample of the previous analysis interval (TFIN) and the current value of pitch (TPLC). For unvoiced frames (voicing decision TVDS=0), the location of the interval is fixed in the frame and determined from the constant scalar TUVL (scalars TFIN and TUVL must be contiguous). The samples in the long-term-DC-removed input sample buffer TSDCH that are within the pitch-synchronous interval are preemphasized with a factor of 0.9375, and the short-term DC value is computed and removed from

them. These samples are stored in the output buffer TSAN. A mean squared energy term is computed over a subset of the samples in TSAN. The number and location of the samples in TSAN used for this energy computation is also determined from the pitch (i.e., a pitch-synchronous interval is used), and the energy is coded and stored in the three-frame RMS energy buffer TRMS. TNFR is the first of four contiguous scalars used in the above computations.

COVMX(TPHI,TSAN,TPSI)

This function (written by CSPI and modified by BBN) computes the covariance matrix (TPHI) and the correlation vector (TPSI) from the buffer TSAN.

MSFBS(TRCO,TPSI,TPHI,0)

This function (written by CSPI and modified by BBN) computes a set of pseudo-reflection coefficients (TRCO) from the modified covariance matrix (TPHI) and correlation vector (TPSI).

2.3.1.8 End of Function List

MPEFL(ANLZA/B)

This CSPI function specifies the end of the indicated function list.

2.3.1.9 Error Protection and Bitstreaming

The EPROL module takes as input a TSNK buffer (either TSNKA or TSNKB) containing quantized and coded analysis parameters and produces as output a TBITS buffer (again, TBTA or TBTB according to the TSINK buffer designation) in which:

1. for frames in which the voicing decision is "unvoiced" or "transition", the 4 MSB's of the gain and first four reflection coefficients are protected by (8,4) Hamming codes, which correct a single-bit error and detect a double-bit error; the parity bits of these error-protecting codes are transmitted in place of reflection coefficients five through ten;

2. the data to be transmitted has been "bitstreamed", one bit per half-word, in the form used by the TMODEM scroll program;

3. histogram information of the coded analysis parameters has been recorded.

These operations are performed in the CSPU, and therefore they can be executed concurrently with an AP operation.

The format of the TSINK buffer is shown in Table 3, along with the number of bits per parameter.

Word	Parameter	No. of bits
0	Pitch/voicing	7
1	RMS	5
2-5	RC1-RC4	5
6-9	RC5-RC8	4
10	RC9	3
11	RC10	2

TABLE 3. TSINK BUFFER FORMAT (LPC-10)

The TBITS buffer contains one data bit in the rightmost bit of each 16-bit half-word. The format of the TBITS buffer (also the frame of data transmitted) is shown in Table 4. (Bits of coded parameters are numbered starting with bit 0 on the right.)

A histogram-gathering function was included in the EPROL module for gathering statistics on the effectiveness of the quantization tables for the analysis parameters and for verifying correct transmitter operation. A simple Fortran program (HISTLPC) was implemented to read the histogram buffers defined on Bus 1 and list them in a text file.

2.3.2 LPC-10 Receiver

The LPC-10 Receiver is, with one exception, identical in organization to the APC/SQ Receiver, which is illustrated in Fig. 4. Unlike the APC/SQ Receiver, which has two speech output buffers (RSNKA and RSNKB) and status flags (RSNFA and RSNFB), the LPC-10 Receiver has three of each (ROS0, ROS1, ROS2 and ROS0F, ROS1F, ROS2F). This is necessitated by the pitch-synchronous nature of the LPC-10 speech synthesis algorithm, which fills the output buffer only up through the last complete pitch period that can be contained therein; the remainder is filled as part of the first pitch period of the following frame. This means that in a given frame, two output buffers must be available to the synthesis process while a third is being emptied by the D/A output process. Section 2.5.2.2.3 describes further consequences, at the control level, of this triple output buffer.

The coded bitstream input, frame synchronization, and D/A

BIT	VOICED& UNVOICED DATA	BIT	VOICED DATUM	UNVOICED DATUM
0	K1-0	27	K2-4	K2-4
1	K2-0	28	K7-0	K3-5
2	K3-0	29	K8-0	R-5
3	P-0	30	P-4	P-4
4	R-0	31	K4-4	K4-4
5	K1-1	32	K5-0	K1-5
6	K2-1	33	K6-0	K2-5
7	K3-1	34	K7-1	K3-6
8	P-1	35	K10-0	K4-5
9	R-1	36	K8-1	R-6
10	K1-2	37	K5-1	K1-6
11	K4-0	38	K6-1	K2-6
12	K3-2	39	K7-2	K3-7
13	R-2	40	K9-0	K4-6
14	P-2	41	P-5	P-5
15	K4-1	42	K5-2	K1-7
16	K1-3	43	K6-2	K2-7
17	K2-2	44	K10-1	NOT USED
18	K3-3	45	K8-2	R-7
19	K4-2	46	P-6	P-6
20	R-3	47	K9-1	K4-7
21	K1-4	48	K5-3	K1-8
22	K2-3	49	K6-3	K2-8
23	K3-4	50	K7-3	K3-8
24	K4-3	51	K9-2	K4-8
25	R-4	52	K8-3	R-8
26	P-3	53	SYNC	SYNC

P = PITCH, R = RMS, K = REFL. COEFF., LSB = BIT 0
LSB OF PARITY BITS = BIT 5

TABLE 4. TBITS BUFFER FORMAT (LPC-10)

output portions of the Receiver are virtually identical to the corresponding portions of two real-time speech coders previously developed by BBN, and they are described in the final reports documenting those coders [1, Vol. II; 8]. This section describes the synthesis processing portion of the Receiver.

The SYNTHESIZE module implements the LPC-10 synthesis algorithm described in [9,12]. Six versions of the SYNTHESIZE module exist, one for each combination of the two input buffers and three output buffers required for proper buffering. Each version consists of a function list of MAP300 function calls. The six versions (SYNA01, SYNA12, SYNA23, SYNB01, SYNB12, SYNB23) are identical in terms of the sequence of functions called and differ only in the parameters (buffers and/or scalars) passed to several of the functions. The SYNA01 module does synthesis from input buffer RSRA into output buffers ROS0 and ROS1 (and unbitstreams from RBTB into RSRB); the names of the other synthesis modules have similar interpretations.

As in the case of Section 2.3.1, this section describes, on a function by function basis, the synthesis function lists, which also appear in subroutine LPC10F.

2.3.2.1 Unbitstreaming, Error Correction, Parameter Smoothing, and Decoding

The ECORDL module takes as input an RBITS buffer containing a frame of bitstream data input from the modem and produces as output an RPAN/RPBN buffer containing error-corrected, smoothed, and decoded floating point values of synthesis parameters, ready for immediate use by the speech coder synthesizer.

The format of the RBITS buffer is the same as the TBITS buffer, shown in Table 4 above.

Unvoiced and transitional frames transmit only four reflection coefficients, and these values and the gain parameter are heavily protected by Hamming (8,4) codes. Voiced frames, on the other hand, transmit ten reflection coefficients, and have no room to spare for such error-protection. (In both cases, the pitch/voicing parameter is transmitted as a 7-bit codeword that includes error-protection.)

Because of these differences between unvoiced/transitional and voiced frames, many of the bits in the received frame are interpreted differently depending on the received (and error-corrected) pitch/voicing. Voiced frames are processed by parameter-smoothing heuristic algorithms, which alleviate the effects of channel errors by median-smoothing parameters that

exhibit unreasonable deviations from the frames on either side [11, 13]. This smoothing is performed on the coded parameters. Then the parameters are decoded by lookup in tables of floating-point values. In addition, a set of reflection coefficients interpolated between the current and previous frames is produced for use in the subsequent epoch-specification step; this interpolation is carried out in the log area ratio domain, by means of converting the coded reflection coefficients to log area ratio coefficients before interpolation and decoding.

As in the case of the error-protection and bitstreaming operation, this operation is performed in the CSPU, and its execution is done in parallel with an AP module.

2.3.2.2 Begin Function List

MPBFL(SYNA/B01/12/23)

This CSPI function specifies the start of the indicated function list.

2.3.2.3 Clear G-flag

MPGSC(IG3,ICLR)

This BBN function causes general purpose flag G3 to be cleared, indicating the start of synthesis processing. This use of flag G3 is intended for system debugging, timing, and internal measurement, and is not required for proper operation of the speech coding software.

2.3.2.4 Synthesis Frame History Buffer Updates

RHISTB(RPAP/RPBP,RPBN/RPAN,ROS0,ROS3) (SYNx01 only)
RHISTC(RPAP/RPBP,RPBN/RPAN) (others)

These BBN functions perform frame history updates on one or two buffers. The previous frame's "current" predictor coefficients from RPBN/RPAN are moved to become the current frame's "past" predictor coefficients RPAP/RPBP. Also, for the SYNA01/SYNB01 function lists only, the previous frame's partial output buffer ROS3 is "wrapped around" into the current frame's partial output buffer ROS0.

2.3.2.5 Reflection to Predictor Coefficient Conversion

PCOEF2(RPANC/RPBNC,RPANC/RPBNC,RPAIC/RPBIC,RPAIC/RPBIC)

This BBN function converts the current and interpolated reflection coefficients, RPANC/RPBNC and RPAIC/RPBIC respectively, to predictor coefficients (in place).

2.3.2.6 Epoch Generation

MPWT(PRCSR,AP)
EPOCHL(REPT,REX,RPAN/RPBN)

These functions (MPWT is a CSPI function; EPOCHL is a BBN function) produce the specifications for the pitch-synchronous speech synthesis, using the decoded frame parameters RPAN/RPBN as input. The MPWT delays the start of EPOCHL (which is a CSPU function) until after the completion of the PCOEF2 module in the AP. Two output buffers are produced: REPT is the "epoch table", which contains for each pitch-synchronous "epoch" to be synthesized, a set of synthesis parameters (duration, gain, predictor coefficients, and position in the buffer). REX contains the excitation signal for the synthesis filtering operation.

2.3.2.7 Pitch-Synchronous Synthesis

SYNTHL(ROS1/2/3,RPHH,REPT,RLR,RSY,ROS0/1/2F,REX)

This BBN function performs the pitch-synchronous speech synthesis that is specified by the REPT (epoch table) and REX (excitation) buffers, followed by deemphasis. The output is put into the partially filled output buffer from the previous frame (ROS0/1/2) and the current frame's output buffer (ROS1/2/3). After the synthesis is complete, the appropriate buffer status flag (ROS0/1/2F) is set to 1 to indicate that the output buffer is full. Scalars RPHH and RLR hold the deemphasis history sample and the previous epoch's RMS value. Buffer RSY is a scratch buffer.

2.3.2.8 Unbitstreaming, Error Correction, and Decoding

ECORDL(RPBN/RPAN,RBTFB/A,RBTB/A)

This BBN function performs the unbitstreaming, error correction, parameter smoothing, and decoding operations described in Section 2.3.2.1 above. The bitstream input is the RBTB/A buffer, and the parameter output is the RPBN/RPAN buffer. The bitstream buffer status flag RBTFB/A is cleared, indicating that the associated RBITS buffer can now be considered empty (i.e., ready to be filled with new received bitstream data).

2.3.2.9 End of Function List

MPEFL(SYNA/B01/12/23)

This CSPI function specifies the end of the indicated function list.

2.4 SYSTEM HARDWARE

The speech coder systems are implemented on a MAP-300 array processing computer, which is manufactured by CSP Inc., of Billerica, Mass. Section 2.4.1 describes the configuration of MAP-300 equipment that is necessary for the speech coder system. Section 2.4.2 describes the additional audio signal interface and IOS-2SM scroll modifications for connection to a modem that complete the system.

2.4.1 MAP-300 Hardware

The MAP-300 configuration that was specified by Defense Communications Agency for the implementation of the speech coder system is listed below.

1	1030 MAP-300 Processor
1	2030 8Kx32 MOS Master Memory, 500 nsec, Bus 1
1	2050 16Kx32 MOS Slave Memory, 500 nsec, Bus 1
1	2203 8Kx32 MOS Master Memory, 300 nsec, Bus 2
1	2410 4Kx32 MOS Master Memory, 170 nsec, Bus 3
1	3110 PDP-11 Interface
1	4020 Model 2SM I/O Scroll
2	4040 Bus Switch (for Model 2SM I/O Scroll)
1	5120 Analog Data Acquisition Module
1	5130 Analog Output Module
1	6100 Expansion Chassis
1	6200 Auxiliary Power Supply

2.4.2 Audio and Modem Interface Hardware

In addition to the MAP-300 equipment listed above, two other pieces of equipment enable the MAP-300 to function as a complete, stand-alone speech coder system: an audio signal interface and a digital data interface to a modem. These two items were designed and built by the GTE Sylvania Electronic Systems Group in Needham, Mass. (see Appendix A of [1]). Identical interfaces were provided for all three DCA MAP-300 systems so that they would be interchangeable at the hardware level.

The audio signal interface consists of a handset, tape input and output jacks, and circuitry for the amplification, equalization, and filtering necessary for interfacing speech input and output signals to the MAP-300 A/D and D/A converters.

The modem interface consists of modifications to the MAP-300 IOS-2SM scroll processor for the purpose of transferring data from MAP memory to the modem and also data from the modem into MAP memory. Two real-time clocks derived from a single master oscillator are also provided for controlling the modem data rate and the speech sampling rates.

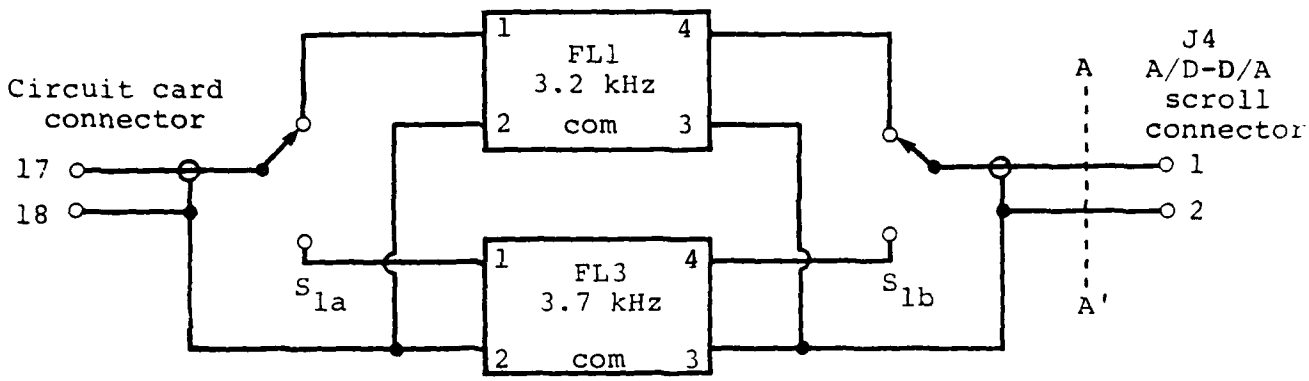
2.4.3 New 100 Hz High-Pass Filter

The APC/SQ and LPC-10 algorithms specify 100 Hz high-pass

filtering of the input speech signal prior to digitization [9,10,12]. Because the lack of such filtering could affect algorithm performance, particularly when the input speech is corrupted by low frequency noise, the audio interfaces of all three DCA MAP-300 systems were modified to include a 100 Hz high-pass filter in the speech input circuit, along with a DPDT switch for switching it in or out of the circuit. The filter is manufactured by TT Electronics of Los Angeles, Calif. (Model H356, -3 dB point = 100 Hz, source and load impedances = 3000 ohms). Figures 6A and 6B show the modified input circuit. Because the filter matches the circuit impedance, its passband insertion loss is negligible, and no extra attenuation is necessary when the filter is switched out of the circuit. Figure 7 shows the high-pass skirt response measured from the filter together with the skirt response described in [9].

2.5 SYSTEM SOFTWARE

Each speech coder contains two distinct sets of software modules. The first set is made up of modules that run in the MAP-300, including CSPI-supplied as well as BBN-written programs. Section 2.5.1 describes the MAP-300 modules that must be loaded into the MAP-300 processor before the speech coder can be operated.

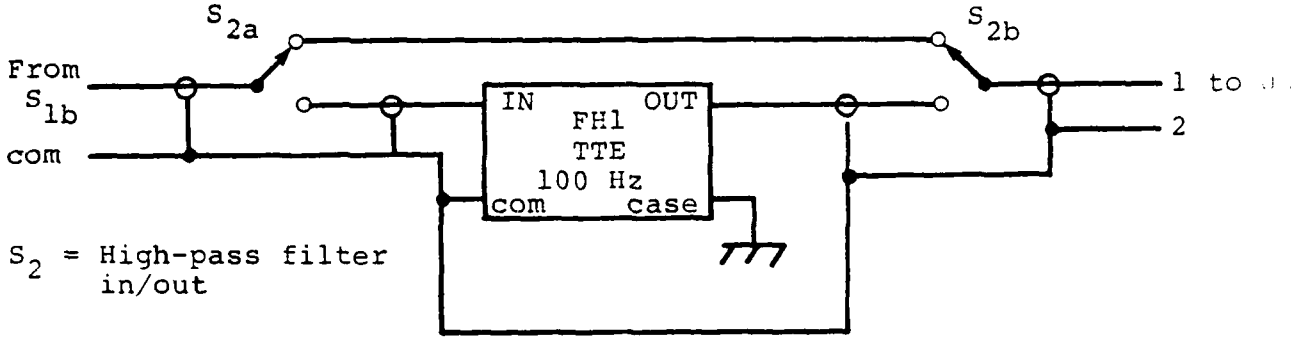


S_1 = Low-pass filter select

FIG. 6A. ORIGINAL INPUT FILTERING CIRCUIT.

The second set of software modules consists of FORTRAN routines that run in the host computer. These routines make use of the CSPI-supplied SNAP-II software system to initialize and start the MAP-300 speech coder programs. Section 2.5.2 describes these routines.

A single basic software structure is used for both the APC/SQ and the LPC-10 implementations. Unless otherwise indicated, the information in this section applies to both systems.



Note: Disable C25 on circuit card by shorting it.

FIG. 6B. HIGH-PASS FILTER AND SWITCH, INSERTED AT A-A' IN FIG. 6A

2.5.1 MAP-300 Software Components

All MAP-300 processing is done in conjunction with Release 3.5 of the CSPI-supplied SNAP-II software system. For the most part, BBN-written MAP-300 programs take the form of new AP or CSPU functions, callable via the standard SNAP-II calling procedure. In addition, BBN-written interrupt service routines and input/output scroll programs have been added to the executive.

CSPI-supplied MAP-300 software necessary for speech coder

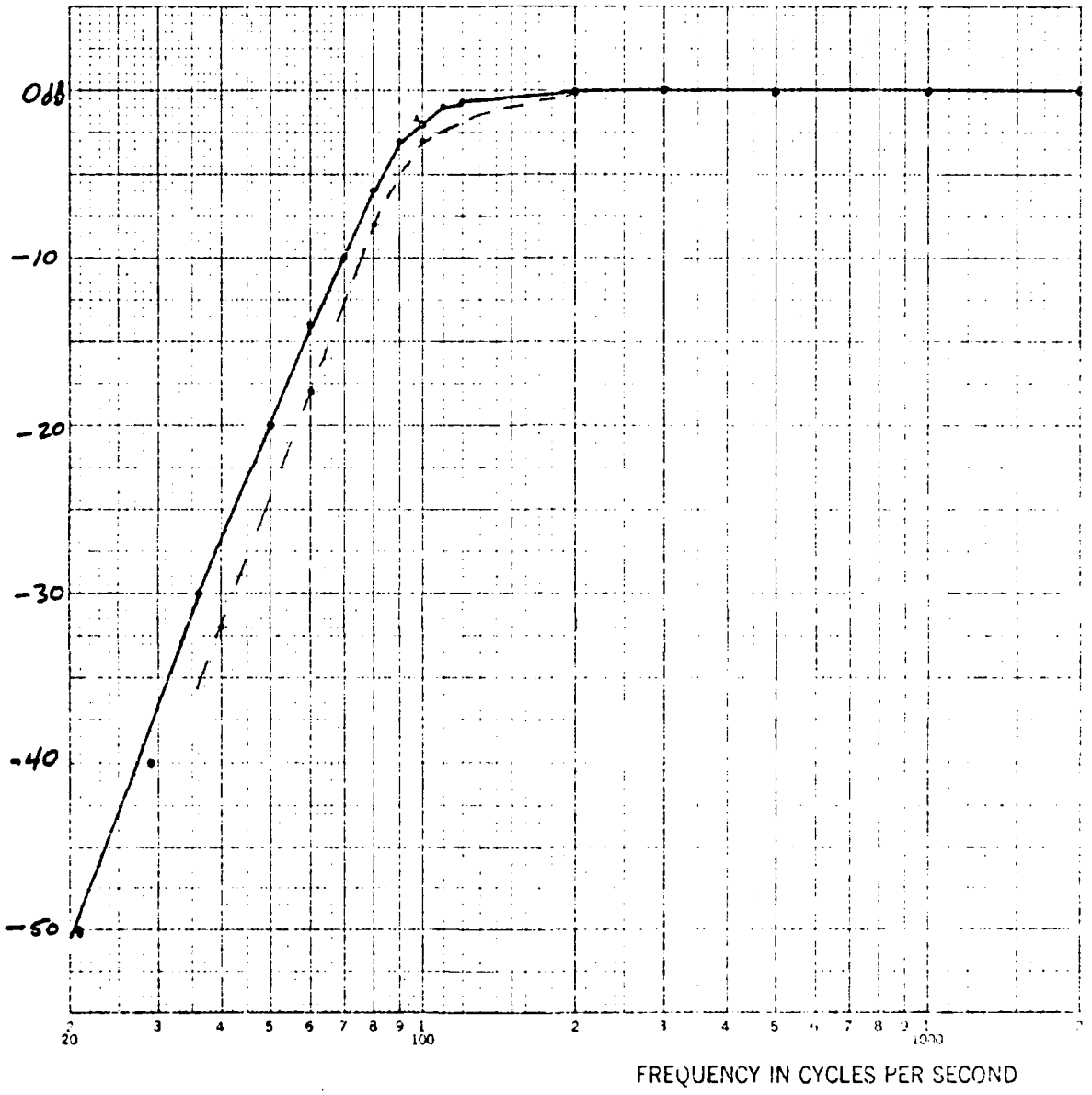


FIG. 7. SKIRT RESPONSES OF TTE HIGH-PASS FILTER (SOLID LINE) AND THE HIGH-PASS FILTER DESCRIBED IN [9], (DASHED LINE).

operation is described in Section 2.5.1.1. MAP-300 speech coder software written by BBN is described in Section 2.5.1.2.

Figures 8 and 9 show the MAP-300 memory organizations for the APC/SQ and LPC-10 coder implementations, including the location of CSPI and BBN software, as well as the location of buffers defined in each speech coder system.

2.5.1.1 CSPI-Supplied MAP-300 Software

The following CSPI-supplied MAP-300 software modules are required for speech coder operation:

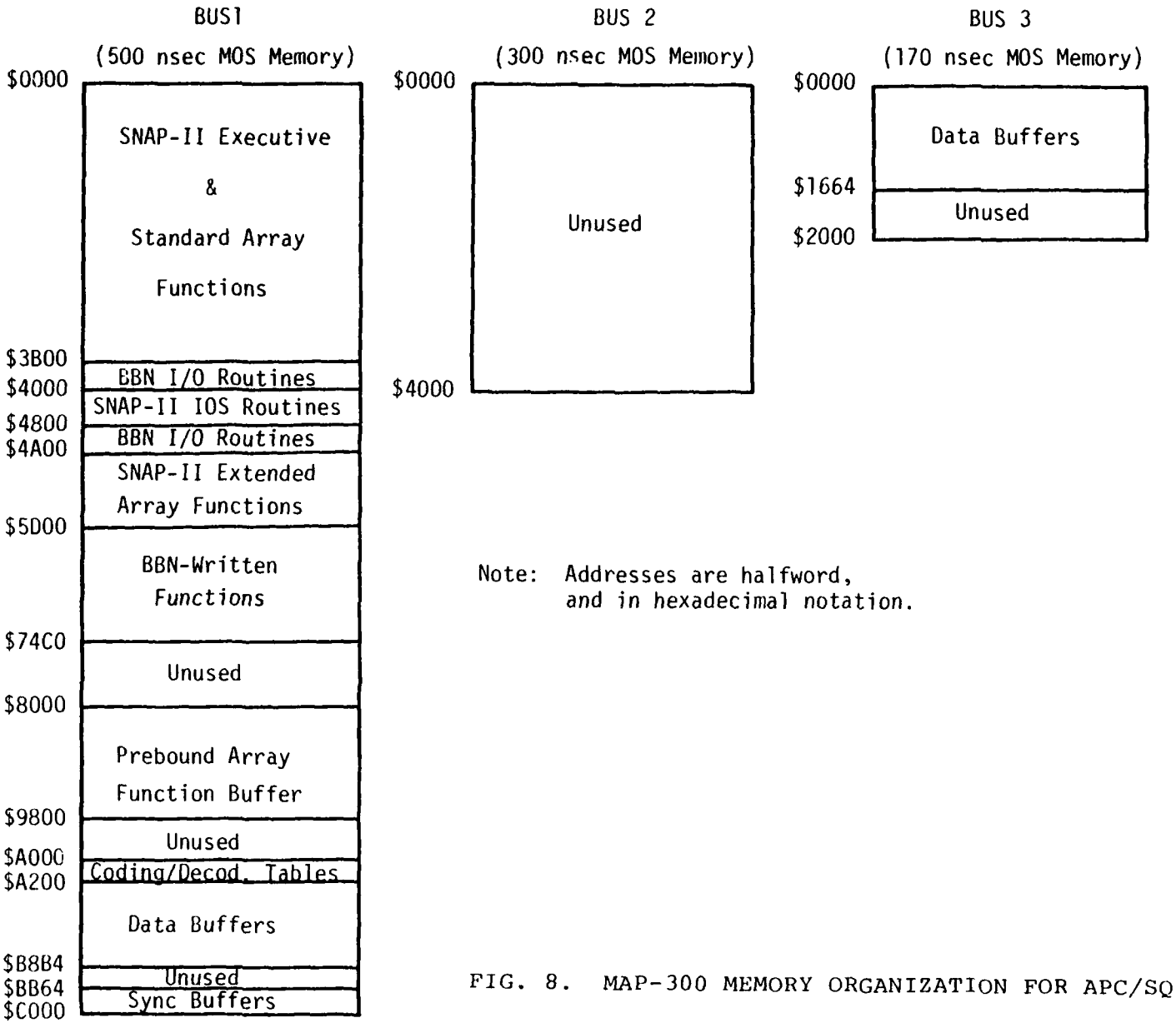
SNAP-II Software System Release 3.5
Model 8300-RSX11M.
(This package includes the SNAP-II Executive and the Standard and Extended Array Functions.)

SNAP-II Input/Output Scroll Package
Release 0.1 Model 8400-RSX11M.
(This package includes the SNAP-II IOS Modules.)

These software modules are described in CSPI documentation [2,3,4,5].

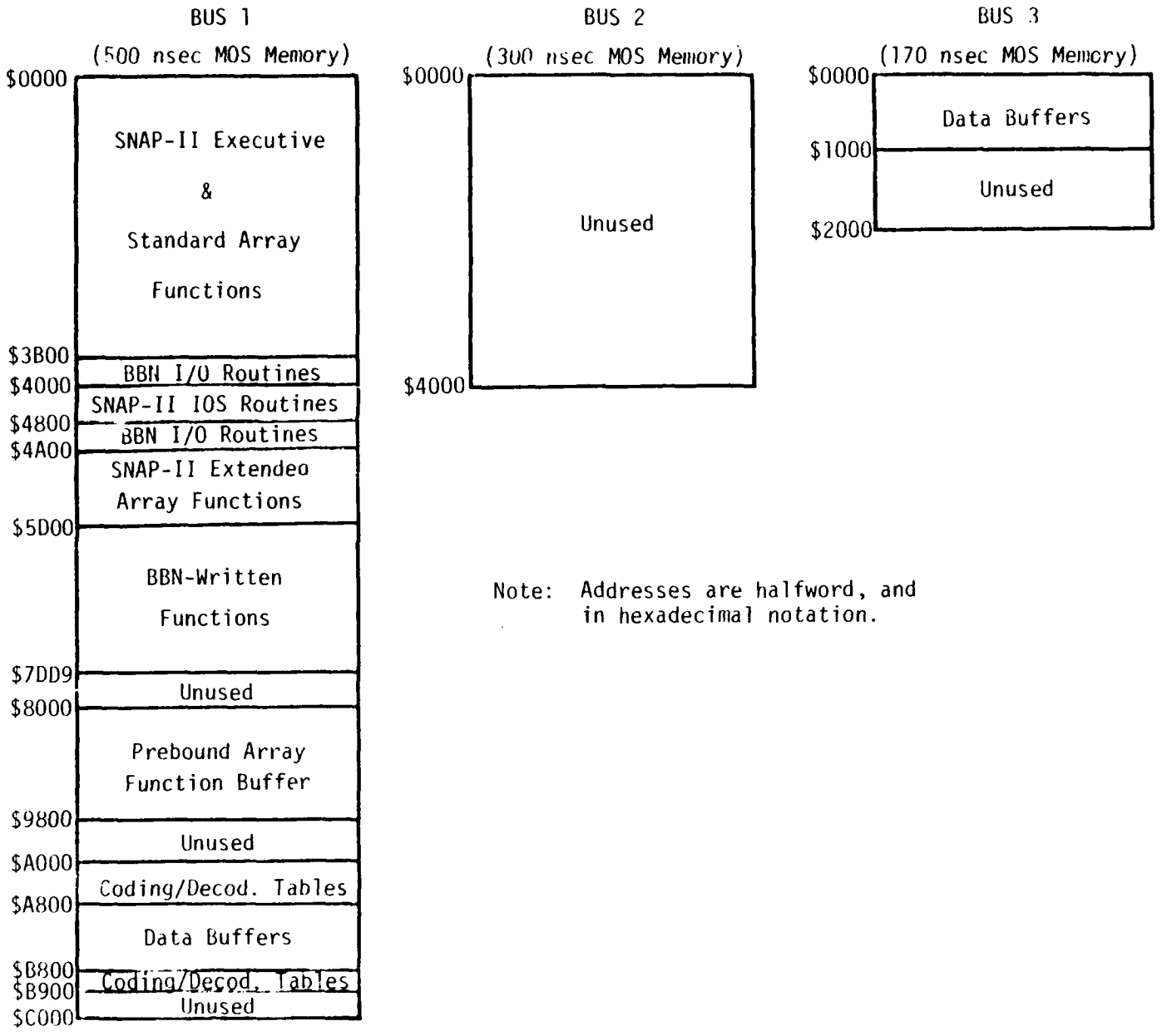
2.5.1.2 BBN-Written MAP-300 Software

BBN-written additions and modifications to the SNAP-II software system for each coder are contained in four separate files. (For all BBN-written MAP-300 modules, a file extension of



Note: Addresses are halfword, and in hexadecimal notation.

FIG. 8. MAP-300 MEMORY ORGANIZATION FOR APC/SQ



Note: Addresses are halfword, and in hexadecimal notation.

FIG. 9. MAP-300 MEMORY ORGANIZATION FOR LPC-10

.MSO designates "MAP source", .MOB designates "MAP object", and .MLI designates "MAP listing".) The files with first name "APCSQM" and "LPC10M" contain added SNAP-II functions (array and non-array) for algorithmic and system support processing. "APCSQT" and "LPC10T" contain tables for quantization and decoding. "APCSQU" and "LPC10U" contain the error protection and correction modules, ADAM, AOM, and IOS-2 programs, and CSPU routines that respond to interrupts from these devices. "APCSQP" and "LPC10P" contain patches to Release 3.5 of the SNAP-II executive.

BBN-written SNAP-II functions are functionally described in Appendices B and C of this report. Each function has been assigned a Function Control Block (FCB) number. An entry has been made for each new FCB in the Function Dispatch Table (FDT), indicating the location of the APU, APS, and/or CSPU module(s) that implement the associated function.

Several unused CSPI-supplied functions, normally available with Release 3.5 of the SNAP-II software system, have been disabled to provide room in the FDT for BBN-written functions. The following SNAP-II functions are unavailable once the BBN-written MAP-300 APC/SQ software has been loaded:

CVMUL	(FCB# 176)	VRAMP	(FCB# 185)
CCVML	(FCB# 177)	CSMA1	(FCB# 188)
CVRCP	(FCB# 178)	CMML	(FCB# 220)

CPOL	(FCB# 179)	CMINV	(FCB# 224)
CSDOT	(FCB# 180)	MWLD	(FCB# 225)
CRECT	(FCB# 181)	ADMRB	(FCB# 229)
CXMUL	(FCB# 182)	VHIST	(FCB# 230)
CXMRL	(FCB# 183)	VRAN1	(FCB# 235)
VPOLY	(FCB# 184)		

The following SNAP-II functions are unavailable once the BBN-written MAP-300 LPC-10 software has been loaded:

CVMUL	(FCB# 176)		
CCVML	(FCB# 177)	CMML	(FCB# 220)
CVRCP	(FCB# 178)	CMINV	(FCB# 224)
CPOL	(FCB# 179)	MMLD	(FCB# 225)
CXMUL	(FCB# 182)	ADMRB	(FCB# 229)
CXMRL	(FCB# 183)	VHIST	(FCB# 230)
VPOLY	(FCB# 184)	VRAN1	(FCB# 235)

These functions can be made available by reloading the SNAP-II software system without loading BBN-written MAP-300 software.

In addition, the standard SNAP-II interrupt routines for interrupts from Device 16 (Lines 1 and 2), Device 22 (Line 1), and Device 23 (Line 1) have been modified to transfer control to BBN-written interrupt routines. The standard interrupt routines for these devices can be similarly accessed by reloading the SNAP-II Executive without loading BBN-written MAP-300 software.

2.5.2 Host Computer Software Components

The proper operation of the speech coder is defined and controlled by a group of programs running in the host computer.

These programs include a set of SNAP-II support subroutines and a MAP-300 driver, supplied by CSPI, as well as a number of FORTRAN programs written by BBN that make use of the CSPI-supplied routines to perform the specific task of defining and controlling the operation of the speech coder in the MAP.

CSPI-supplied host computer programs are described in Section 2.5.2.1. Host computer programs written by BBN are described in Section 2.5.2.2.

2.5.2.1 CSPI-Supplied Host Computer Software

The following CSPI-supplied host computer software modules are required for speech coder operation:

SNAP-II Software System Release 3.5
Model 8300-RSX11M.
(This package includes the SNAP-II Host Support Packages or Standard and Extended Array Functions.)

SNAP-II Input/Output Scroll Package
Release 0.1 Model 8400-RSX11M
(This package includes the SNAP-II IOS Host Support package.)

DEC RSX-11M I/O Driver Model 8901

These software modules are described in CSPI documentation [2,3,5,6,7].

2.5.2.2 BBN-written Host Computer Software

A set of BBN-written FORTRAN programs performs the multiple tasks of defining and initializing MAP-300 buffers and scalars and of defining and executing the sequence of functions in the MAP that perform the actual speech coding operations. These FORTRAN programs are organized around a single mainline program (APCSQR/LPC10R), which calls, in turn, subroutines to configure MAP buffers and scalars (APCSQC/LPC10C), initialize these buffers and scalars (APCSQI/LPC10I), define function lists for various speech coder tasks (APCSQF/LPC10F), and interact with the user in controlling the execution of these function lists (APCSQE/LPC10E). These subroutines are described in Sections 2.5.2.2.1 through 2.5.2.2.4 below.

2.5.2.2.1 Buffer and Scalar Configuration (APCSQC/LPC10C)

The task of defining MAP-300 buffers and scalars to the SNAP-II executive is performed by subroutine APCSQC/LPC10C. Upon entry, this routine first initializes the SNAP-II executive via the MPOPN function. All SNAP-II buffers are then configured using the MPCLB function, with buffer ID numbers, sizes, and addresses defined symbolically within the routine. In general, transmitter buffer names begin with "T", while receiver buffer names begin with "R". Buffer sizes are stored in variables named

by prefixing an "S" to the the buffer name. Similarly, buffer addresses are stored in variables named by prefixing "A" to the buffer name. All real scalars and integer scalars are then defined by assigning scalar ID numbers to the associated symbolic scalar names. Transmitter scalar names generally begin with "T", while receiver scalar names begin with "R".

All buffer and scalar names (and certain buffer sizes and addresses) are included in FORTRAN labeled COMMON blocks, which permit all other FORTRAN subroutines in the speech coder system to reference these symbolic buffer and scalar names in their calls to SNAP-II functions.

2.5.2.2.2 Buffer and Scalar Initialization (APCSQI/LPC10I)

MAP-300 buffers and scalars are set to their initial values by subroutine APCSQI/LPC10I. Only certain buffers and scalars require such initialization.

2.5.2.2.3 Control Structure Definition (APCSQF/LPC10F)

Subroutine APCSQF/LPC10F defines several function lists that specify the operation of the speech coder system. Function list ID numbers are defined symbolically. The FORTRAN variables containing the function list ID numbers are included in a labeled COMMON block so that they can be referenced by other subroutines.

As part of the function list definition task, this subroutine produces pre-bound versions of all of the SNAP-II array functions used in the speech coder system. Pre-binding is a SNAP-II operation that causes function parameter information, normally communicated to the function at execution time, to be "bound" to the function at some earlier time (in this case at system initialization time). Approximately 6100 half-words are required on Bus 1 to store the pre-bound versions of the speech coder array functions.

Function lists are defined for the analysis and synthesis sections of the speech coder. Since the systems are multiple-buffered, the analysis and synthesis function lists are each defined twice (six times for LPC-10 synthesis) to allow for the different sets of input and output buffers. These function lists specify the sequence of MAP-300 array and non-array functions, operating on the system buffers and scalars previously defined, that implement the analysis and synthesis processes described in Sections 2.2 and 2.3 of this report.

The analysis and synthesis function lists described above are used in the definitions of several other function lists that define the higher-level structure of the speech coder systems. Function list APCLP/LPCLP defines the real-time speech coder loop, executing analysis or synthesis function lists as

appropriate. The APC/SQ and LPC-10 analyzers and the APC/SQ synthesizer are double-buffered on both input and output, and a simple data-flow principle suffices for determining when to execute each analysis or synthesis function list: execute it if and only if the buffer status flags indicate that its input buffer is full and its output buffer is empty. The MPIFF function calls in the APCLP/LPCLP function lists have precisely this effect. The LPC-10 receiver, however, has six synthesizer function lists to handle all combinations of the 'A' and 'B' input (RBITS) buffers and the '0', '1', and '2' output (ROS) buffers. The data-flow principle used above no longer suffices for proper execution of the LPC-10 receiver; not only must the input buffer be full and two of the three output buffers be empty, but execution must cycle through the output buffers in sequence. The SYNSEL function calls in the LPCLP function list have precisely this effect.

Function list APCRTS/LPCRTS defines the start-up sequence for the real-time speech coder system, starting the Modem, ADAM, and AOM Scroll processors and repeatedly executing the APCLP/LPCLP function list. This repeated execution is conditioned on the contents of MAP integer scalar RUN being non-zero; hence, the speech coder can be stopped by setting RUN to zero. Function list APCRST/LPCRST restarts the real-time speech coder by setting RUN to non-zero and reinitiating the repeated execution of the real-time speech coder loop.

Two other function list definitions are included in this subroutine to support non-real-time file-to-file speech coder operation, speech coder timing operation, and speech coder oscilloscope display operation (for detailed internal timing). These modes of speech coder operation are not supported, and can not be invoked, in the delivered speech coder system. They were used for system development and debugging use under the VAX/VMS operating system and are included here to aid in future additions to the speech coder system. Function list APCFFT/LPCFFT specifies the speech coder loop with explicit calls to functions that execute the A/D, RMODEM, TMODEM, and D/A interrupt routines. Function list APCTIM/LPCTIM invokes an ANLZA and SYNZA pair for approximate system timing purposes.

All SNAP-II functions are called via host computer support subroutines that perform the actual communication with the MAP driver. These subroutines are contained in library SNPLIB for CSPI-supplied functions and in file APCSQH/LPC10H for BBN-written functions.

2.5.2.2.4 System Software Execution (APCSQE/LPC10E)

The execution of the speech coder system is controlled by subroutine APCSQE/LPC10E. This subroutine starts the real-time speech coder system by first loading the Modem, ADAM, and AOM

Scrolls with their respective programs, and then executing the real-time speech coder start-up function list (APCRTS/LPCRTS) described in the preceding section. It then interacts with the user, responding to single-character commands to halt the speech coder ('Q'), enable/disable error simulation ('E'), enable/disable error correction ('C'), cause the speech coder to lose sync artificially ('L'), type out a group of speech coder state scalars ('T'), and suspend the controlling host computer task, allowing the speech coder to continue executing in the MAP-300 ('S').

Provision is included in this subroutine for the user to select other speech coder modes of operation, specifically file-to-file, timing, and oscilloscope display modes. However, this mode selection process is bypassed in the delivered speech coder system, and real-time speech coder execution mode is forced. As described in the previous section, these other operating modes are not supported, and cannot be invoked, in the delivered speech coder system. They were used for system development and debugging and are included here to aid in future additions to the speech coder system. File APCSQD contains dummy versions of various system-dependent subroutines, which constitute the unsupported software portions of these unsupported modes of speech coder operation.

2.6 SYSTEM TIMING PERFORMANCE

2.6.1 APC/SQ System Timing

The APC/SQ speech coder system introduces a total delay of 9 frames, corresponding to 225.0 milliseconds, between voice input and synthesized voice output (not including any delays in transmission). This consists of 4 frames of delay introduced by the transmitter portion of the system and 5 frames of delay contributed by the receiver portion. In the transmitter, a frame of delay each is produced by the buffering in the ADAM scroll program, the ANALYZE processing module, and the TMODEM scroll program. In addition, the ANALYZE module introduces a frame of "folding" delay due to the concurrent execution of the ANALYZE module and the PROPAR/PRRES3/PRRES1 modules (See Section 2.2.1).

In the receiver, two frames of delay are contributed by the buffering in the RMODEM scroll program, and a frame each is produced by buffering in the SYNTHESIZE processing module and the AOM scroll program. In addition, the SYNTHESIZE module introduces a frame of "folding" delay due to the concurrent execution of the SYNTHESIZE module and the CORPAR and DECRES modules (see Section 2.2.2).

The observed internal timing of the APC/SQ speech coder modules is shown in Fig. 10. This figure reflects data obtained

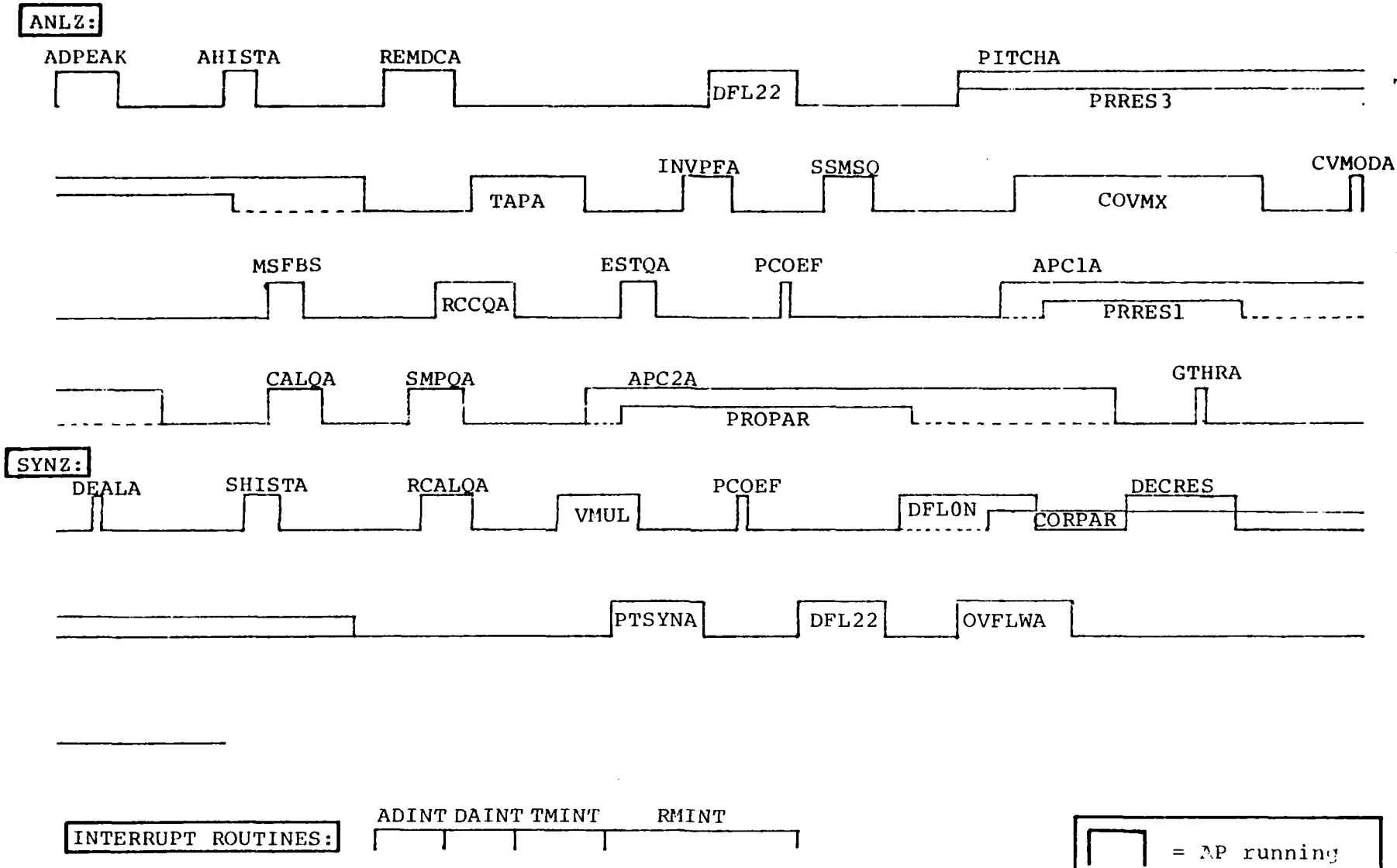


Figure 10. APC/SQ MAP-300 Timing.

by running the speech coder in a loop without the ADAM, AOM, and MODEM scroll processors and observing the states of the MAP-300 RA flag (APU run) and G-flags with an oscilloscope. The speech coder was provided with simulated A/D input data. ADAM, AOM, and MODEM scroll interrupt routines were omitted from the timing loop. These interrupts were timed separately, and their execution times are indicated at the end of the figure. Including these interrupt routines, Fig. 10 shows a total processing time of 23.87 milliseconds for a 25.0 millisecond frame of data, or about 0.955 times real time. (This is the worst case total processing time. If the ADAM, AOM, or MCDEM scroll interrupts were to occur while the AP were running and the CSPU were idle, some or all of the interrupt routine execution time would be hidden behind the concurrent AP routine.)

2.6.2 LPC-10 System Timing

The LPC-10 speech coder system introduces a total delay of 13 frames, corresponding to 292.5 milliseconds, between voice input and synthesized voice output (not including any delays in transmission). This consists of 6 frames of delay introduced by the transmitter portion of the system and 7 frames of delay contributed by the receiver portion. In the transmitter, a frame of delay each is produced by the buffering in the ADAM scroll program, the ANALYZE processing module, and the TMODEM scroll

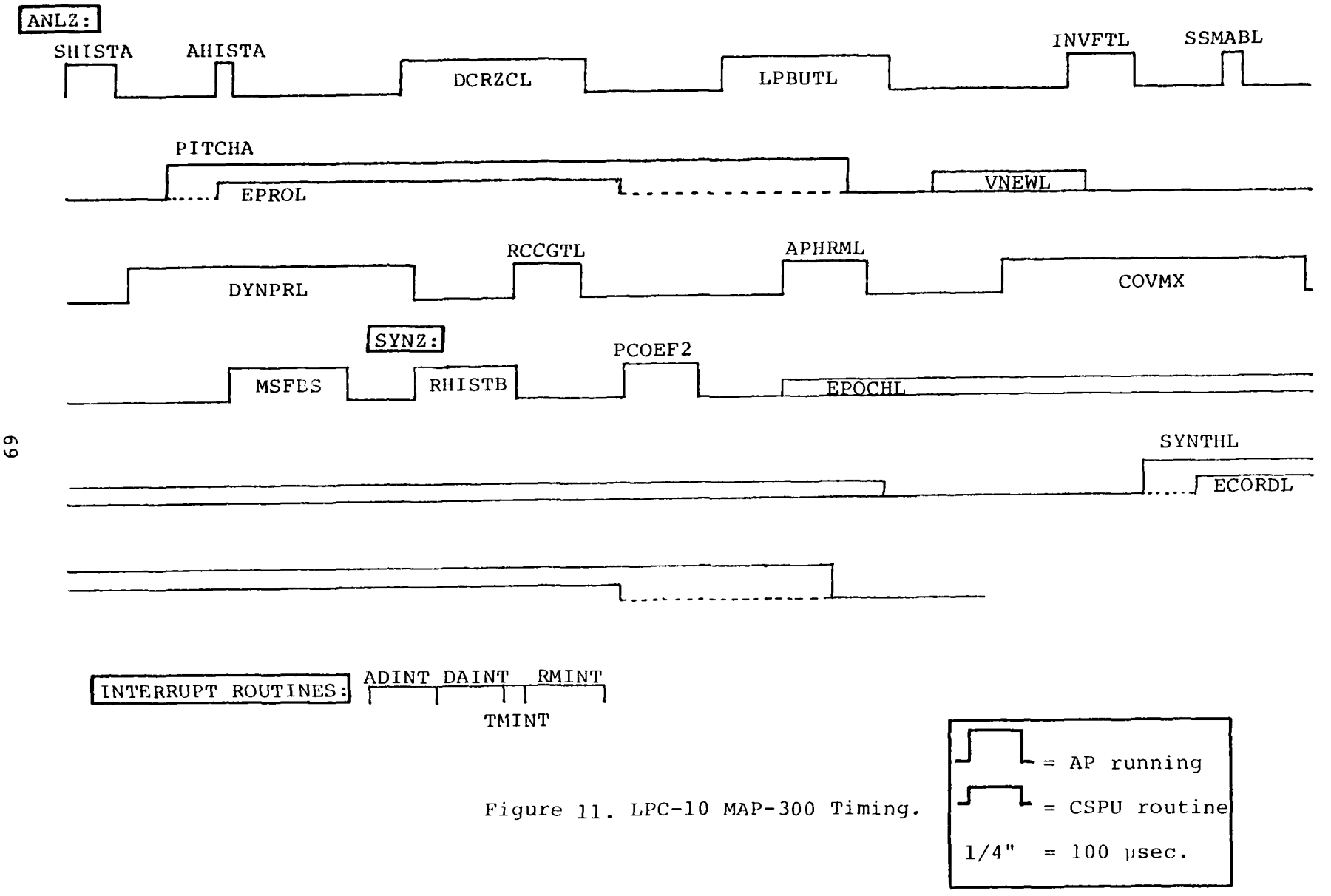


Figure 11. LPC-10 MAP-300 Timing.

program. In addition, the ANALYZE module introduces a frame of "folding" delay due to the concurrent execution of the ANALYZE module and the EPROL modules (see Section 2.3.1) and a further two frames of delay due to the pitch/voicing estimation algorithm.

In the receiver, two frames of delay each are contributed by the buffering in the RMODEM scroll program and in the SYNTHESIZE processing module, and one frame of delay is produced by buffering in the AOM scroll program. In addition, the SYNTHESIZE module introduces a frame of "folding" delay due to the concurrent execution of the SYNTHESIZE module and the ECORDL module (see Section 2.3.2), and the ECORDL module itself introduces a further frame of delay in the parameter-smoothing algorithm.

The observed internal timing of the LPC-10 speech coder modules is shown in Fig. 11. This figure reflects data obtained by running the speech coder in a loop without the ADAM, AOM, and MODEM scroll processors and observing the states of the MAP-300 RA flag (APU run) and G-flags with an oscilloscope. The speech coder was provided with simulated A/D input data. ADAM, AOM, and MODEM scroll interrupt routines were omitted from the timing loop. These interrupts were timed separately, and their execution times are indicated at the end of the figure.

Including these interrupt routines, Fig. 11 shows a total processing time of 22.0 milliseconds for a 22.5 millisecond frame of data, or about 0.98 times real time. (This is the worst case total processing time. If the ADAM, AOM, or MODEM scroll interrupts were to occur while the AP were running and the CSPU were idle, some or all of the interrupt routine execution time would be hidden behind the concurrent AP routine. Furthermore, the pitch-synchronous synthesizer's execution time is strongly dependent on the pitch and voicing. The times shown correspond to the worst case, all voiced with a fundamental frequency of 400 Hz. For unvoiced sounds or for a lower fundamental frequency, the execution time can be as much as 3.5 msec lower.)

2.7 IMPLEMENTATION CORRECTNESS ISSUES

We have taken several steps to confirm the correctness of our MAP-300 real-time implementations. For APC/SQ, we have digitized a number of utterances, by both male and female speakers, at the APC/SQ sampling rate of 7600 Hz, and we have processed the utterances by:

- Our MAP-300 APC/SQ Version 09 implementation, operating in file-to-file mode, and
- The Fortran simulation of APC/SQ Version 09 (a bit-for-bit simulation of the original PFSP implementation, supplied by the Government).

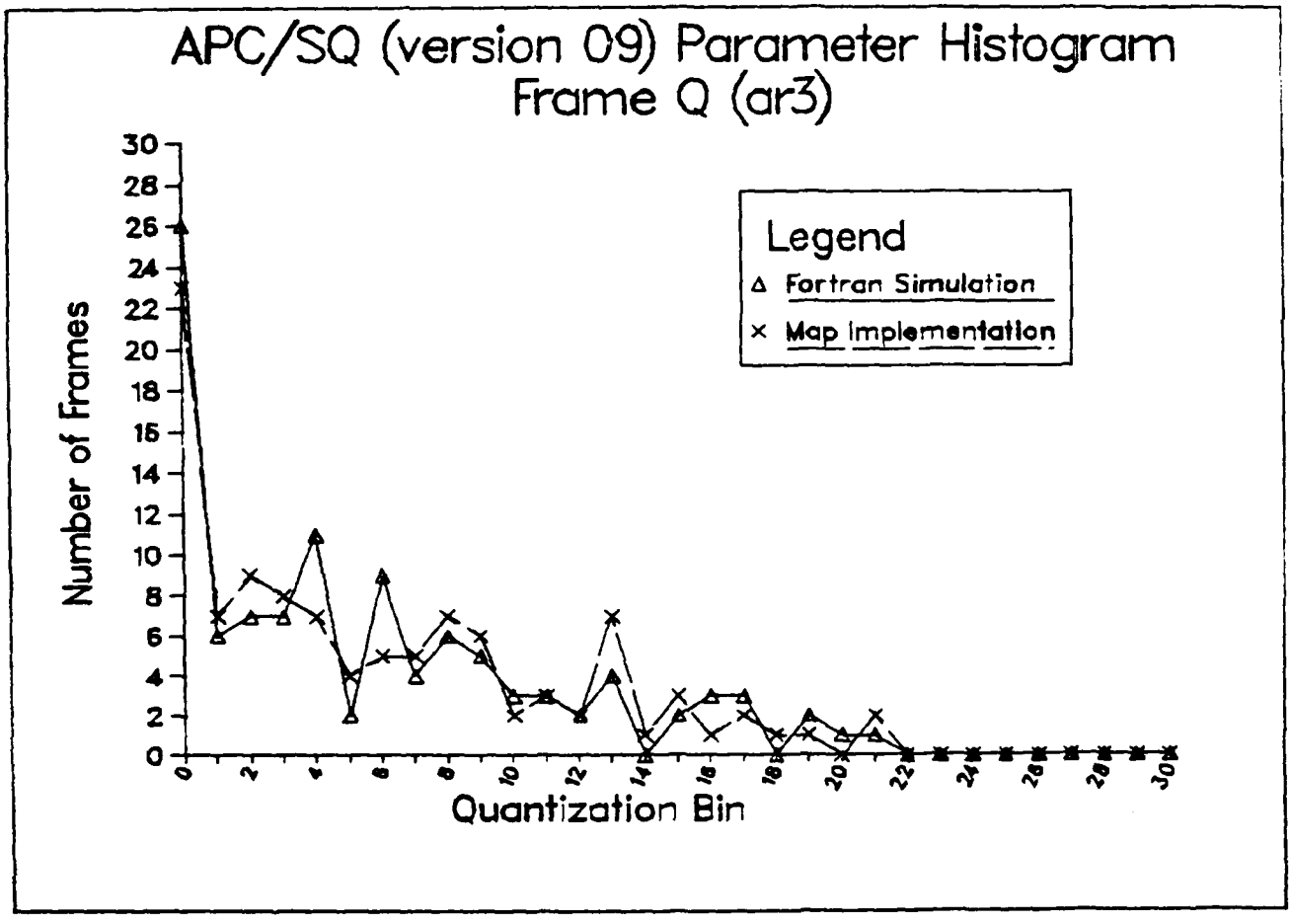


FIG. 12. APC/SQ PARAMETER HISTOGRAM: FRAME Q

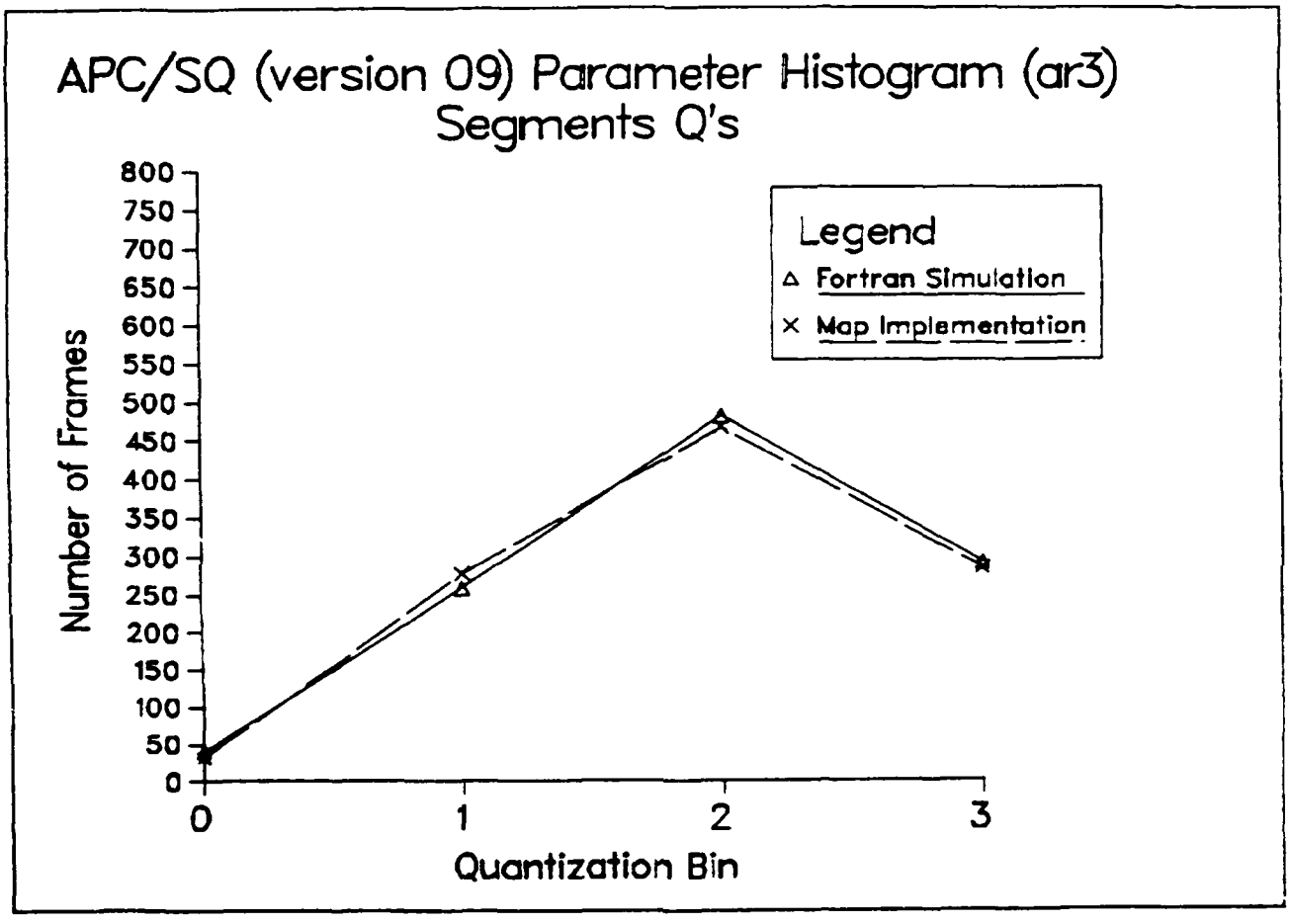


FIG. 13. APC/SQ PARAMETER HISTOGRAM: SEGMENT Q'S

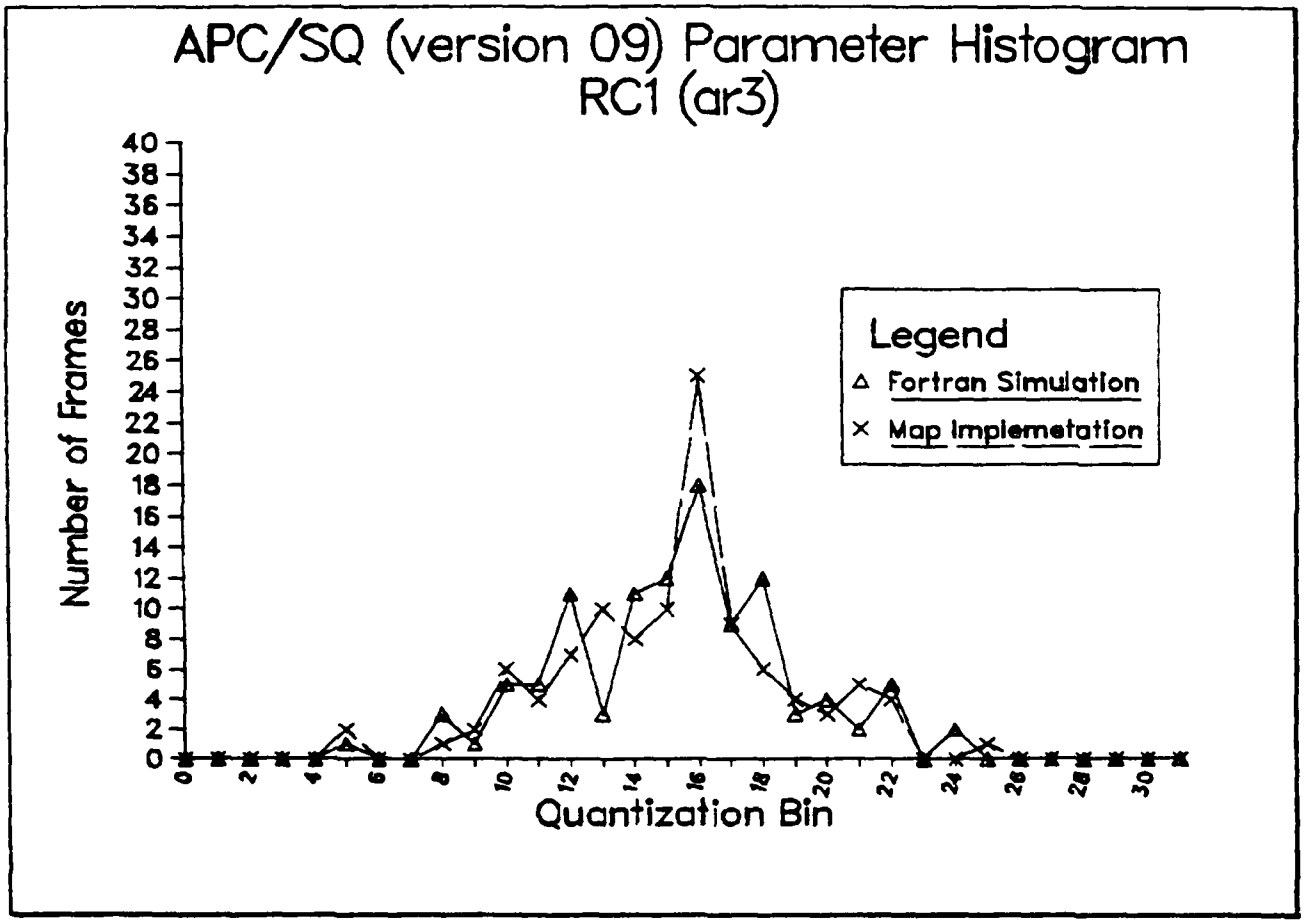


FIG. 14. APC/SQ PARAMETER HISTOGRAM: RC1

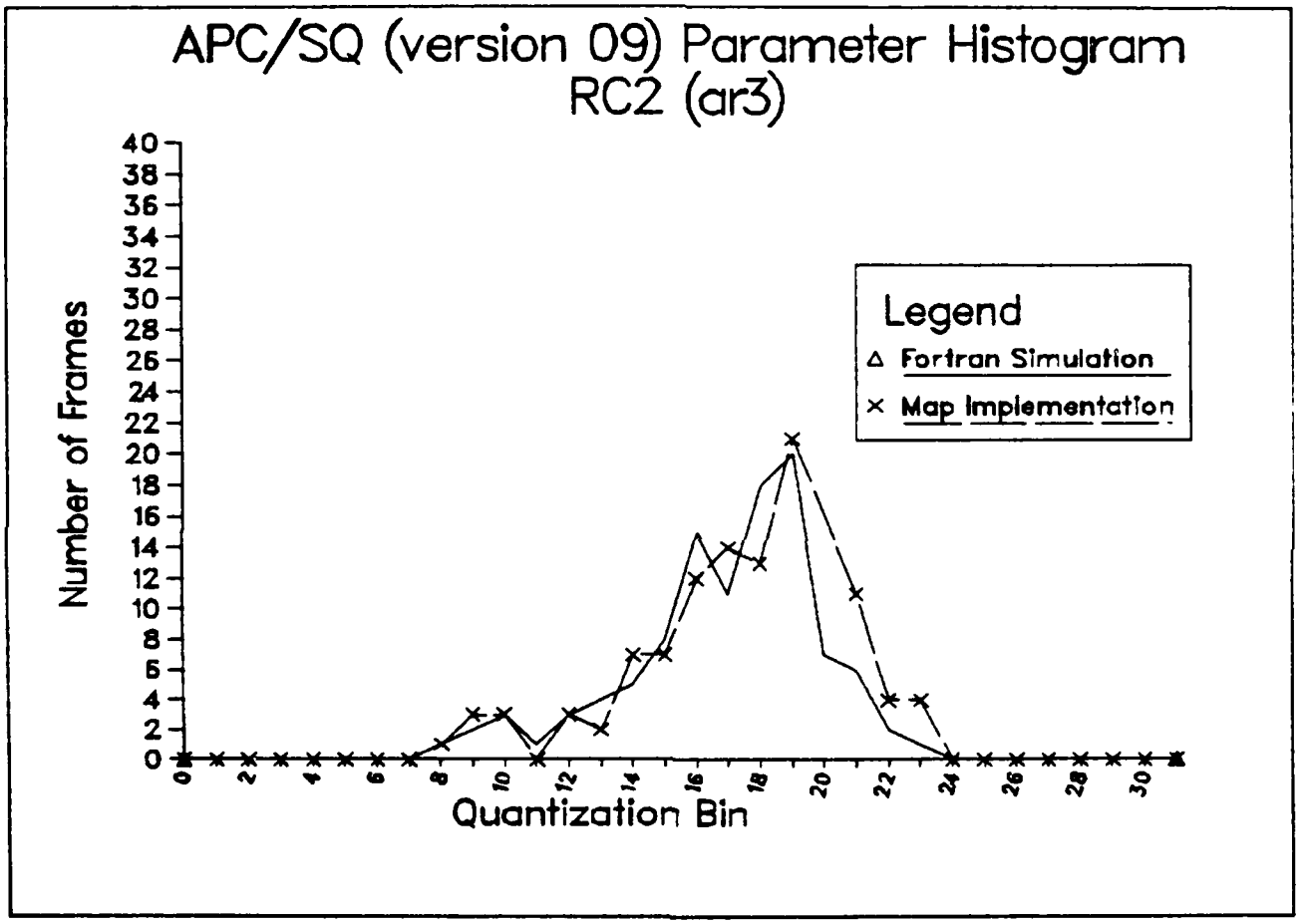


FIG. 15. APC/SQ PARAMETER HISTOGRAM: RC2

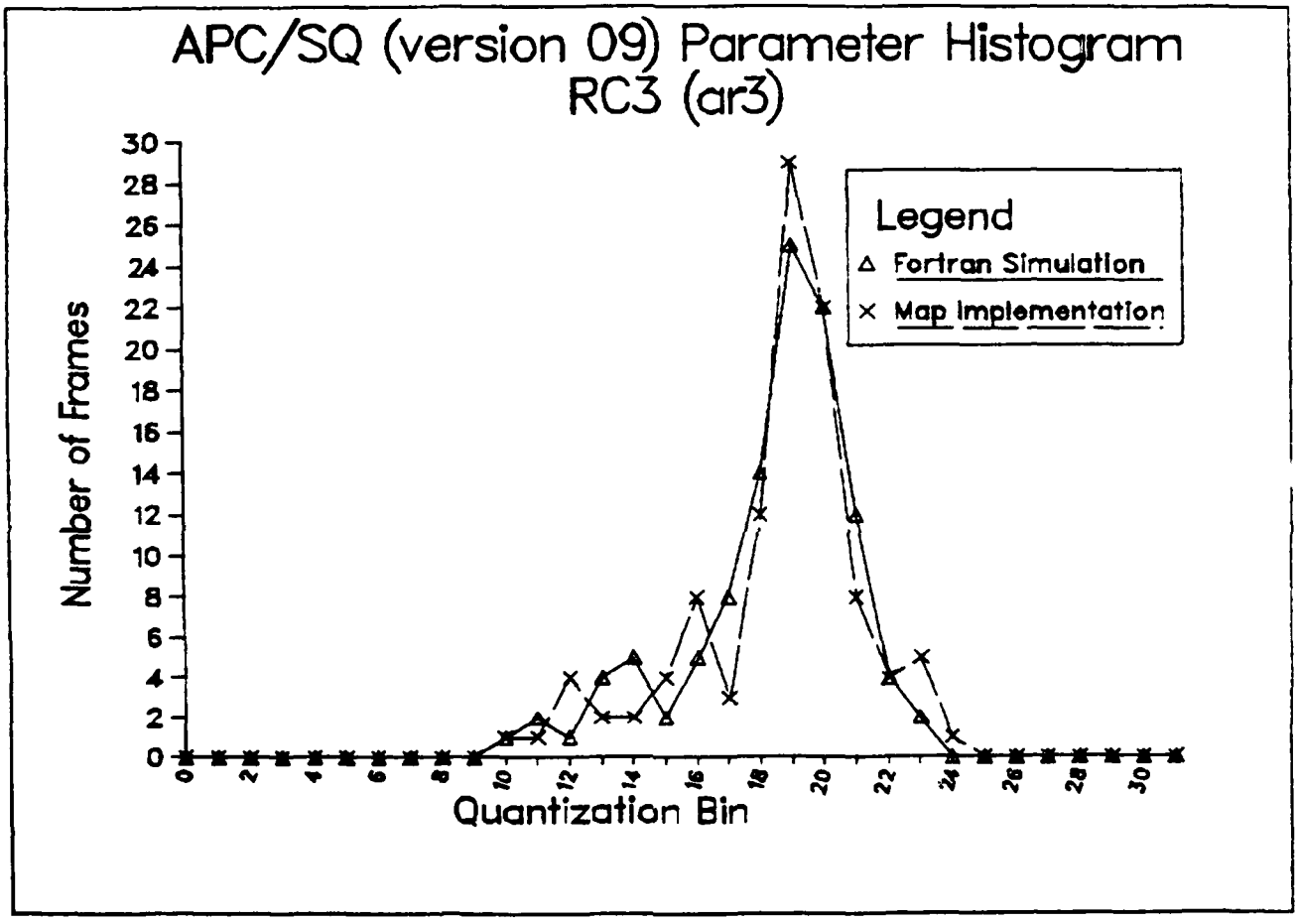


FIG. 16. APC/SQ PARAMETER HISTOGRAM: RC3

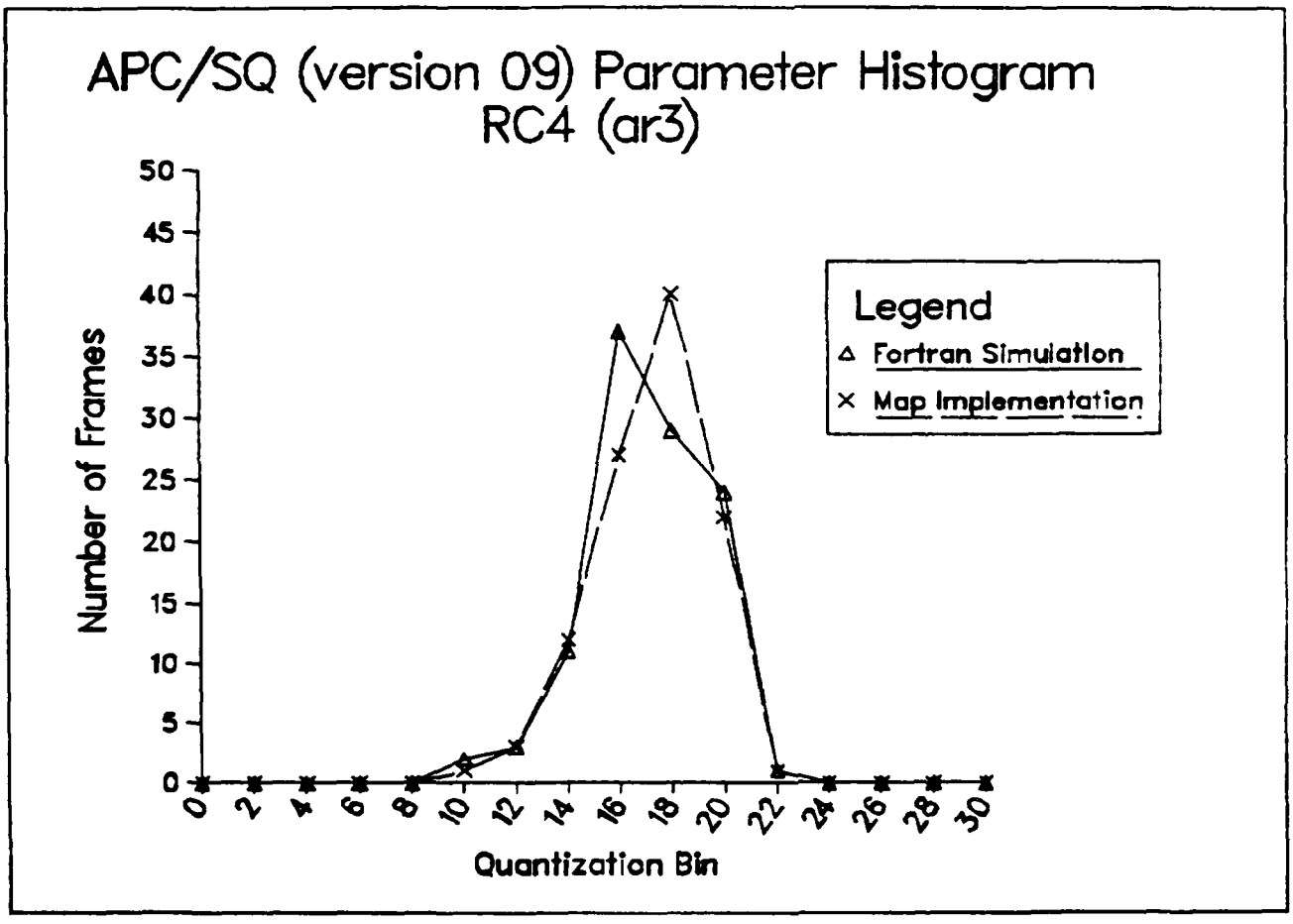


FIG. 17. APC/SQ PARAMETER HISTOGRAM: RC4

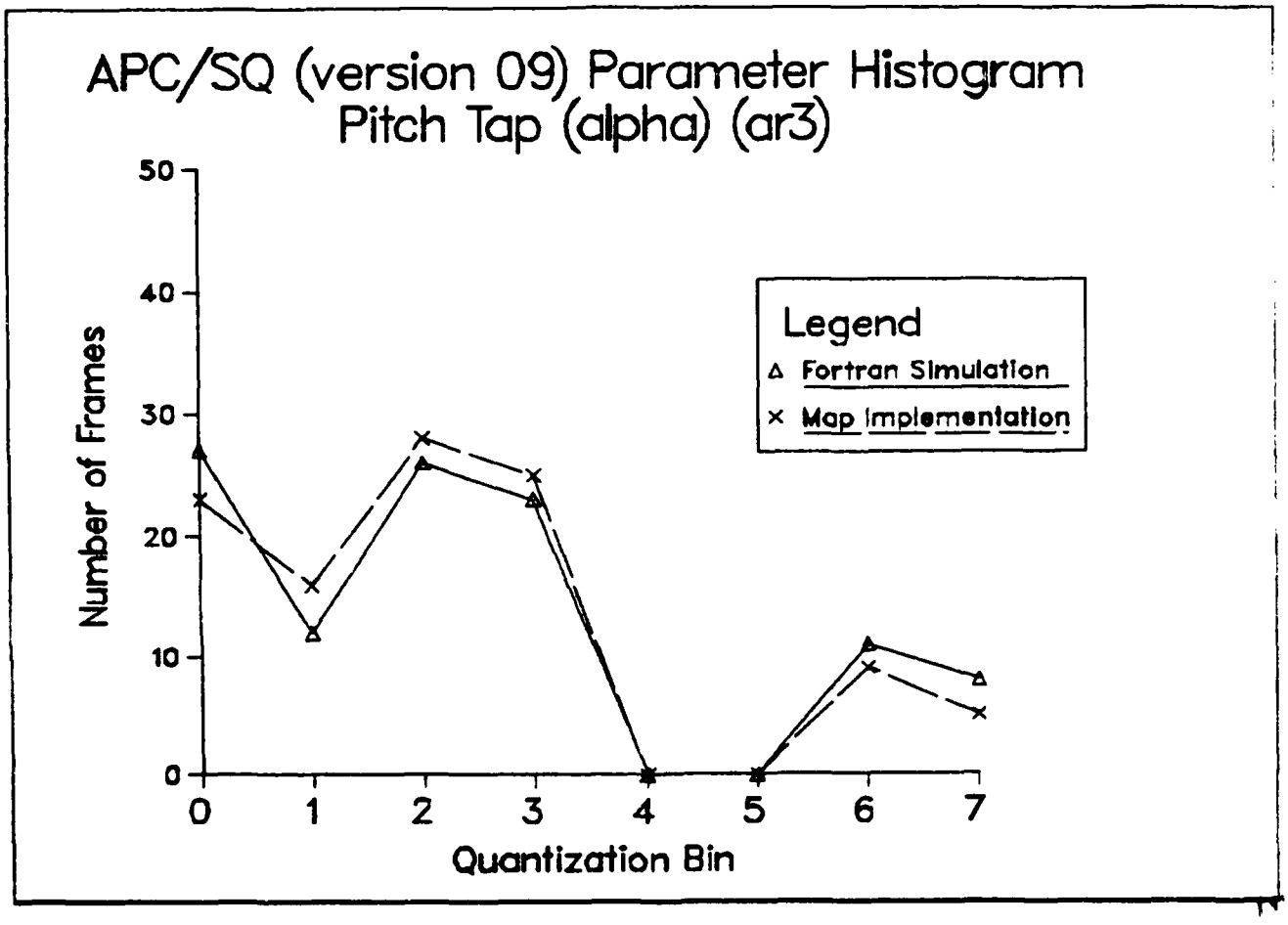


FIG. 18. APC/SQ PARAMETER HISTOGRAM: PITCH TAP

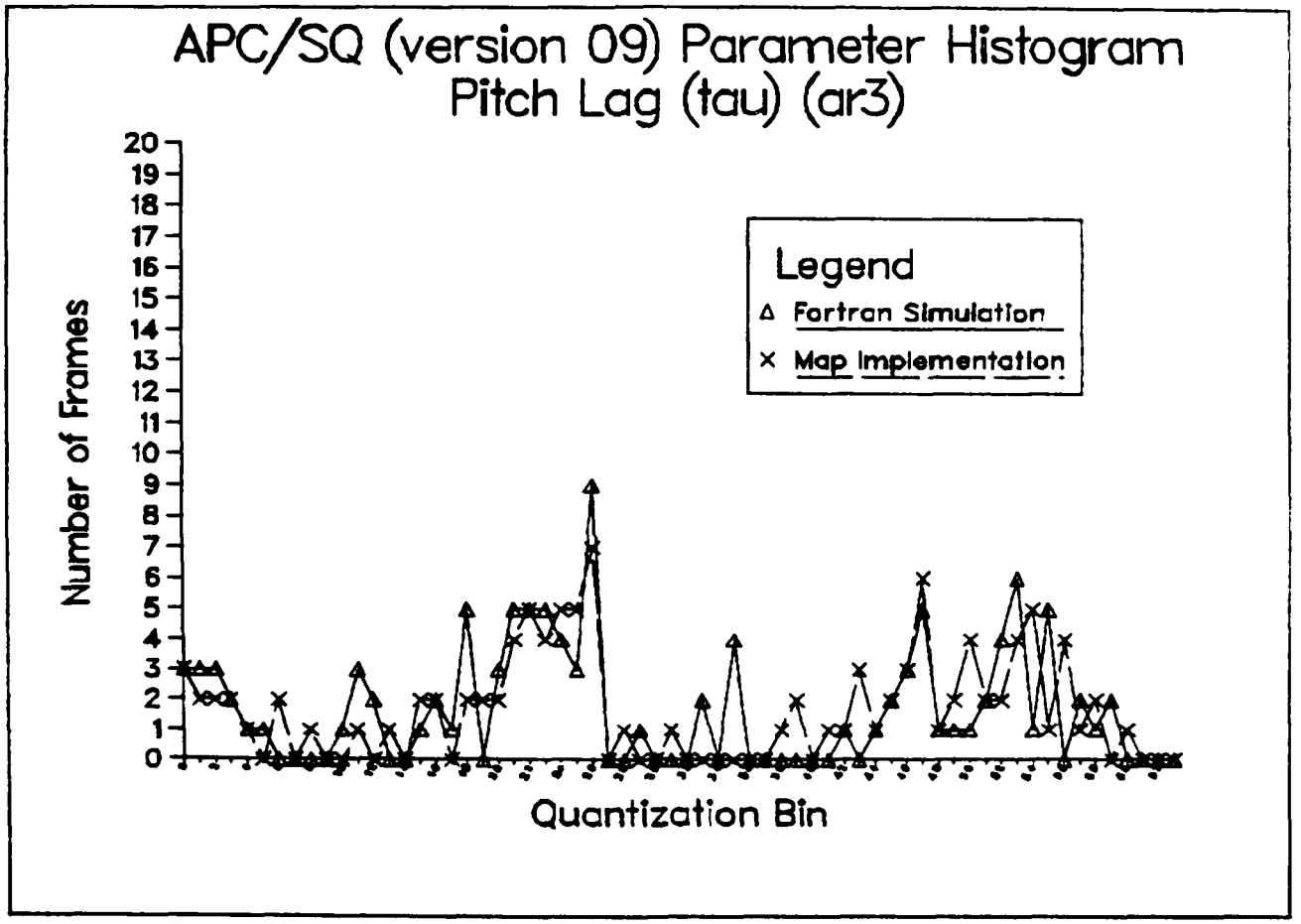


FIG. 19. APC/SQ PARAMETER HISTOGRAM: PITCH LAG

In careful informal listening tests, we have found the two sets of processed utterances to be virtually indistinguishable.

We have also generated and compared sets of histograms of the transmitted APC parameters for the MAP-300 and Fortran implementations of APC/SQ. These are also very similar. Figures 12-19 are graphs of this data for a single utterance (titled "ar3"), comparing the MAP-300 and Fortran data for the eight transmitted APC parameters (all ten Segment Q's are lumped together for this comparison). We attribute the small differences to:

- o The slightly different sampling rate assumed by the MAP-300 implementation (7680 Hz vs. 7600 Hz, as noted in Section 2.2), and
- o The floating point nature of our MAP-300 implementation, as opposed to the fixed point nature of the Fortran simulation.

For LPC-10, an accurate Fortran simulation does not exist (to our knowledge). We have evaluated the correctness of our MAP-300 implementation by:

1. Digitizing a number of DAM test utterances, by both male and female speakers, at the LPC-10 sampling rate of 8000 Hz,
2. Processing them through our MAP-300 LPC-10 Version 44 implementation, operating in file-to-file mode (this involved processing each utterance several times, to allow the adaptive algorithms in the coder to settle to values appropriate for each utterance), and

3. Comparing the processed utterances to the same utterances digitized (at 8000 Hz) from a tape of DAM test material processed by "DoD LPC-10/MRP 2400 BPS (#1336)".

In careful informal listening tests, we have found the two sets of processed utterances to be very similar. We attribute the small differences to:

- o 32-bit floating point computation in our MAP-300 implementation vs. 16-bit integer (plus scaling) computation in the LPC-10/MRP implementation. We have observed differences, particularly at low input signal level, between floating point and integer versions of the same LPC-10 modules.
- o Differences in version between our MAP-300 implementation (Version 44) and the LPC-10/MRP implementation (unknown version).

3. SOFTWARE OPERATING PROCEDURES

The installation and execution of the speech coder system software are accomplished using the RSX-11M operating system as well as MAP-300 support software supplied by CSPI. It is assumed that the RSX-11M operating system, the MAP-300 device drivers, the MAP-300 loaders, and the SNAP-II software system have all previously been installed prior to the speech coder system installation.

After the speech coder system software has been installed, the speech coder can be executed by using the procedure given in Section 3.1. The software installation procedure is given in Section 3.2. The procedures below are specified for the APC/SQ coder. The procedures for the LPC-10 coder are obtained by substituting "LPC10" for "APCSQ" below.

3.1 SOFTWARE EXECUTION PROCEDURE

The speech coder system execution procedure consists of two major steps. First, both MAP-300 processors in the full-duplex speech coder system must be loaded from the host PDP-11 with identical copies of the speech coder MAP-300 software. Second, two RSX-11M tasks must be run to initialize and start the two MAP-300 speech coders.

The following files must be disk-resident before speech coder execution can be attempted:

```
APCSQX.MBN          (MAP Binary file)
APCSQA.TSK          (Host Task for MAP #A)
APCSQB.TSK          (Host Task for MAP #B)
```

If these files do not exist, they must be created according to the speech coder system software installation procedure given in Section 3.2.

The following dialog indicates the proper command sequence for full-duplex speech coder system execution. User commands are underscored to differentiate them from computer responses.

```
>SET /UIC=[6,260]
>ALLOCATE MA:          **allocate MAP #A**
>ALLOCATE MB:          **allocate MAP #B**
>INSTALL APCSQA
>INSTALL APCSQB
>RUN [6,100]QLA        **load MAP #A**
*APCSQX.MBN
TT1 -- STOP

>RUN [6,100]QLB        **load MAP #B**
*APCSQX.MBN
TT1 -- STOP

>RUN APCSQA            **start MAP #A**

APC/SQ 9.6 KB/S SPEECH CODING SYSTEM

CONFIGURING MAP BUFFERS AND SCALARS...
INITIALIZING MAP BUFFERS AND SCALARS...
PREBINDING FUNCTIONS AND DEFINING
FUNCTION LISTS...
EXECUTING APCSQ SPEECH CODER...
SPEECH CODER IS IN OPERATION.
```

```
COMMANDS ARE:
 S:   SUSPEND TASK (LEAVING SPEECH CODER
      RUNNING)
 Q:   QUIT (HALT SPEECH CODE)
 E N: SIMULATE N ERRORS PER FRAME
 C N: ERROR-CORRECT IF N=0
 L:   CAUSE RCVR TO LOSE SYNC
 T:   TYPE OUT SPEECH CODER STATE

ENTER COMMAND: S
APCSQA -- PAUSE (SPEECH CODER CONTINUING)

>RUN APCSQB              **start MAP #B**

APC/SQ 9.6 KB/S SPEECH CODING SYSTEM

CONFIGURING MAP BUFFERS AND SCALARS...
INITIALIZING MAP BUFFERS AND SCALARS...
PREBINDING FUNCTIONS AND DEFINING
FUNCTION LISTS...
EXECUTING APCSQ SPEECH CODER......
SPEECH CODER IS IN OPERATION.

COMMANDS ARE:
 S:   SUSPEND TASK (LEAVING SPEECH CODER RUNNING)
 Q:   QUIT (HALT SPEECH CODER)
 E N: SIMULATE N ERRORS PER FRAME
 C N: ERROR-CORRECT IF N=0
 L:   CAUSE RCVR TO LOSE SYNC
 T:   TYPE OUT SPEECH CODER STATE

ENTER COMMAND: S
APCSQB -- PAUSE (SPEECH CODER CONTINUING)

>
```

At this point in the command sequence, the full-duplex speech coder system should be in operation. Host tasks APCSQA and APCSQB can be resumed (for interaction with the speech coder software) by invoking the RSX-11M "RESUME" command. The speech coder system can be halted either by responding with "Q" to the "ENTER COMMAND:" typeout from APCSQA and APCSQB or by aborting

the APCSQA and APCSQB tasks through the use of the RSX-11M "ABORT" command.

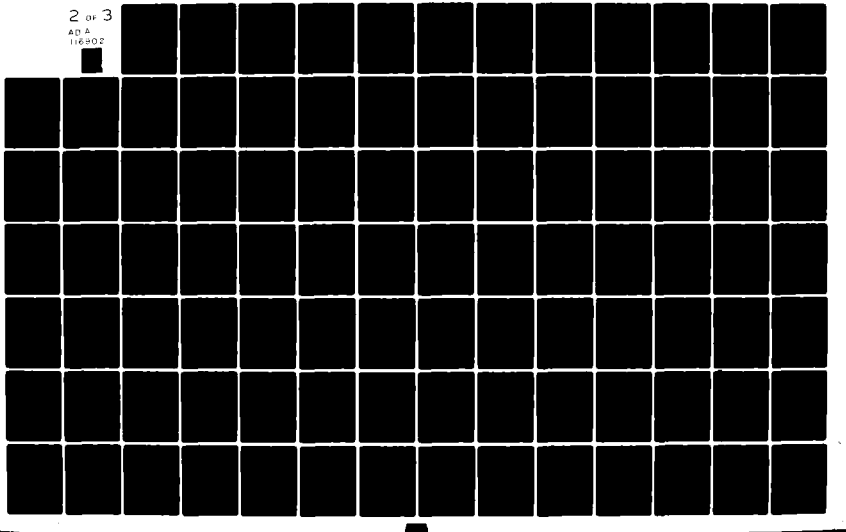
3.1.1 Optional Software Execution Procedure (No G-Flag)

Throughout the APC/SQ Speech Coder MAP software, we have included code to set and clear various general-purpose flags (G-flags). These flags are used mostly for debugging, timing, and system measurement purposes; the manipulations are not required, in general, for speech coder operation. Although these G-flag manipulations are strictly legal and proper according to CSPI documentation, they seem to induce "Bus 1 Memory Timeout" errors after apparently random amounts of time (ranging from a few minutes to several hours). For this reason, we have delivered with the system a patch file (NOGAPC.MOB for APC/SQ, NOGLPC.MOB for LPC10) which, when loaded into the MAP over the APCSQ (or LPC-10) speech coder software, no-op's all G-flag manipulations not needed for speech coder operation. This file can be loaded into the MAP processors (following the load of APCSQX) using the following command sequence:

```
>RUN [6,100]MALD
OBJECT INPUT? NOGAPC.MOB
BINARY OUTPUT?(carriage return)
TT1 -- STOP 2
>RUN [6,100]MBLD
OBJECT INPUT?NOGAPC.MOB
BINARY OUTPUT?(carriage return)
TT1 -- STOP 2
>
```

AD-A116 902 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2
REALTIME IMPLEMENTATION OF THE APC/SQ AND LPC-10 SPEECH CODING --ETC(U)
JUN 82 J J WOLF, K D FIELD, W H RUSSELL DCA100-80-C-0034
UNCLASSIFIED BBN-4855 NL

2 OF 3
AD A
116902



3.1.2 Analog Input Level Setting

The APC/SQ speech coder software includes a facility for monitoring the peak input speech level. This facility allows the level to be set properly when the speech coder is being fed from an adjustable level signal source (such as a tape deck). The peak input level is maintained in integer scalar TADPK. TADPK can be displayed (along with a number of other scalars) during speech coder operation by entering a 'T' in response to the 'ENTER COMMAND:' prompt. The peak input level range is 0-2047, so the desired peak input level is approximately 1500-2000. A peak level above about 2000 introduces a potential input signal clipping condition, while a peak level below about 1500 does not take full advantage of the system's available dynamic range. The 'T' command resets TADPK to 0, so the displayed value pertains only to the time interval since the previous 'T' was typed.

3.2 SOFTWARE INSTALLATION PROCEDURE

The speech coder system software installation procedure consists of two major steps. First, the speech coder MAP-300 software must be converted from MAP object format to MAP binary format. Second, two RSX-11M tasks (for initializing and starting the two MAP-300 processors) must be task-built.

The following files must be disk-resident before speech coder software installation can be attempted (note: file APCSQD has no counterpart in the LPC-10 coder):

```
APCSQX.MOB   (MAP object file)
              (This file consists of the
               following concatenated files,
               with intervening "FFFF" lines
               deleted:
                  SNAP300.MOB
                  IOS.MOB
                  APCSQM.MOB
                  APCSQU.MOB
                  APCSQT.MOB
                  APCSQP.MOB)
APCSQC.FOR   (HOST FORTRAN module)
APCSQD.FOR   (         "          )
APCSQE.FOR   (         "          )
APCSQF.FOR   (         "          )
APCSQH.FOR   (         "          )
APCSQI.FOR   (         "          )
APCSQR.FOR   (         "          )
```

If these files do not exist, they must be recopied to disk from the RSX PIP-format "APC/SQ speech coder s/w" magnetic tape delivered with the speech coder system.

The MAP-300 speech coder software should be converted from MAP object format to MAP binary format using the following command sequence:

```
>RUN [6,100]MALD
OBJECT INPUT?APCSQX.MOB
BINARY OUTPUT?APCSQX.MBN
LOAD MAP?(Y OR N) N
TT1 -- STOP 2
>
```

The two RSX-11M tasks (for initializing and starting the two MAP-300 processors) should be generated next. First, all FORTRAN speech coder modules listed above must be compiled. Then the two tasks must be task-built, with each task including the compiled FORTRAN object modules, the CSPI-supplied SNAP host support library modules, and the appropriate CSPI-supplied MAP device driver for MAP #A or MAP #B. The following command sequence will accomplish this procedure:

```
>F4P APCSOC=APCSOC.FOR/CO:35
>F4P APCSOD=APCSOD.FOR/CO:35
>F4P APCSOE=APCSOE.FOR/CO:35
>F4P APCSOF=APCSOF.FOR/CO:35
>F4P APCSOH=APCSOH.FOR/CO:35
>F4P APCSOI=APCSOI.FOR/CO:35
>F4P APCSOR=APCSOR.FOR/CO:35
>TKB                **build host task for MAP #A**
TKB>APCSOA/FP/CP=APCSOR,APCSOC,APCSOD
TKB>APCSOE,APCSOF,APCSOH,APCSOI
TKB>[6,100]SNALIB/LB
TKB>/
ENTER OPTIONS:
TKB>UNITS=11
TKB>ASG=TI:7
TKB>ASG=MA:10
TKB>//
>TKB             **build host task for MAP #B**
TKB>APCSOB/FP/CP=APCSOR,APCSOC,APCSOD
TKB>APCSOE,APCSOF,APCSOH,APCSOI
TKB>[6,100]SNBLIB/LB
TKB>/
ENTER OPTIONS:
TKB>UNITS=11
TKB>ASG=TI:7
TKB>ASG=MB:10
TKB>//
```

The speech coder system software is now installed. The

speech coder can be executed by using the procedure given in Section 3.1.

4. REFERENCES

1. R. Viswanathan, J. Wolf, L. Cosell, K. Field, A. Higgins and W. Russell, "Design and Real-Time Implementation of a Baseband LPC Coder for Speech Transmission over 9600 BPS Noisy Channels," Final Report, BBN Report No. 4327, Vols. I and II, Bolt Beranek and Newman Inc., Cambridge, MA, February 1980.

2. "Simple Notation for Array Processing, Version II (SNAP-II) Reference Manual," CSPI Document No. JB6000-017-01, C.S.P. Inc., Billerica, MA, November 1978.

3. "Release Note: SNAP-II Release 3 Executive Features and Expanded Array Function Set," CSPI Document No. DW6000-021-04, C.S.P. Inc., Billerica, MA, July 1979.

4. "Release Note: SNAP-II Extended Array Function Library, II," CSPI Document No. RB6000-24-00, C.S.P. Inc., Billerica, MA, March 1980.

5. "SNAP-II Input/Output Scroll Package, User's Manual, CSPI Document No. JW8400-001-01, C.S.P. Inc., Billerica, MA, July 1979.

6. "Installation Procedure for Release 3 SNAP-II Software System on the DEC RSX-11M System," CSPI Document No. LL8901-004-03, C.S.P. Inc., Billerica, MA, July 1979.

7. "MAP Software Interface Description for the DEC RSX-11M System," CSPI Document No. ST8901-000-04, C.S.P. Inc., Billerica, MA, July 1979.

8. J. Wolf, K. Field, W. Russell, "Design and Real-Time Implementation of a Robust APC Coder for Speech Transmission Over 16 Kb/s Noisy Channels, Volume II: Real-Time Implementation," Final Report, BBN Report No. 4565, Bolt Beranek and Newman Inc., Cambridge, MA, Dec. 1980.

9. T. E. Tremain, J. W. Fussell, R. A. Dean, B. M. Abzug, M. D. Cowing, and P. W. Boudra, Jr., "Implementation of Two Real-Time Narrowband Speech Algorithms," Proc. EASCON 1978, Sept. 25-27, 1978, Washington, DC.

10. "Documentation for APC/SQ, Version 08: 9600 BPS Adaptive Predictive Voice Coder With Segment Q," Feb. 1981, unpublished Government document.

11. K. Field, "Documentation for Fortran Simulation of APC/SQ (Version 07)," BBN Report No. 4418, Bolt Beranek and Newman Inc., Cambridge, MA, June 1980.

12. "Documentation for LPC-10, Version 43," Jan. 1981, unpublished Government document.

13. J.W. Fussell, B.M. Abzug, P.W. Boudra, Jr., M.D. Cowing, "Providing Channel Error Protection for a 2400 BPS Linear Predictive Coded Voice System," Proc. 1978 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Tulsa, OK, April 10-12, 1978, pp. 462-465.

APPENDIX A
ALGORITHMIC DESCRIPTION OF APC/SQ

1. INTRODUCTION

This Appendix specifies a floating-point implementation of the APC-SQ (Version 09) voice coding algorithm defined by the Philco Ford Signal Processor (PFSP) fixed point implementation described in [9,10]. A Fortran simulation of the algorithm is described in [11].

This description is oriented toward a floating point implementation on the MAP-300 array processor. There are several major areas in which it differs from the PFSP implementation.

1. Due to hardware limitations, the sampling rate is specified at 7680 Hz, rather than 7600 Hz. At this specified sampling rate, the frame time of 25 msec. produces a frame size of 192 samples (instead of 190 samples). This change in frame size affects the implementation specification in a number of areas, most significantly frame and segment quantizer level calculations and APC loop residual quantization.

2. AGC sample scaling and predictor coefficient scaling has been removed, since the floating point nature of this implementation implies automatic retention of maximal precision and essentially eliminates the possibility of arithmetic overflow.

3. The input and output speech sample range is $-1<x<+1$ instead of $-2048 \leq x \leq 2047$. This affects constants throughout the algorithm, including the decoding table for frame quantizer level.

4. The single-tap pitch filter coefficient is maintained as an unscaled value, instead of being scaled by 0.5.

5. A single sync bit per frame is transmitted, allowing for transmission of all 20 reflection coefficient bits, rather than transmitting two sync bits per frame and only 19 reflection coefficient bits.

2. System Characteristics

- Input and Output speech sampling rate = 7.680 kHz
- Frame Size = 25.0 msec (192 samples)
- Short Term (Spectral) Predictor Order = 4
- Long Term (Pitch) Predictor Order = 1
- Number of Quantizer Segments = 10
- Data Rate = 9600 b/s (240 bits/frame)

Item	Bits/Frame
PITCH LAG	6
PITCH PREDICTOR COEFF	3
REFLECTION COEFFS. (4)	20
FRAME QUANT. SCALE FACTOR	5
SEGMENT Q. SCALE FACTORS (10)	20
ERROR PROTECTION/CORRECTION	5
APC RESIDUAL (180)	180
SYNC	1
	240

- Bit Allocation for Error Protect/Correction

Single (21, 16) modified Hamming error correcting code:

PROTECTED BITS:

PITCH LAG	(6 BITS)
PITCH PREDICT. COEFF.	(1 MSB)
RC1	(1 MSB)
RC2	(2 MSB)
RC3	(2 MSB)
RC4	(2 MSB)
FRAME QUANT. SCALE FACTOR	(2 MSB)
TOTAL	(16 BITS)

3. Analyzer

A data flow diagram of the analyzer is given in Figure 1. In this section, we provide a control flow-chart for each of the various analyzer components. Throughout this section, data buffer/scalar names and processing function names are those used in the MAP-300 implementation of APC/SQ. Arrays are subscripted starting at 1.

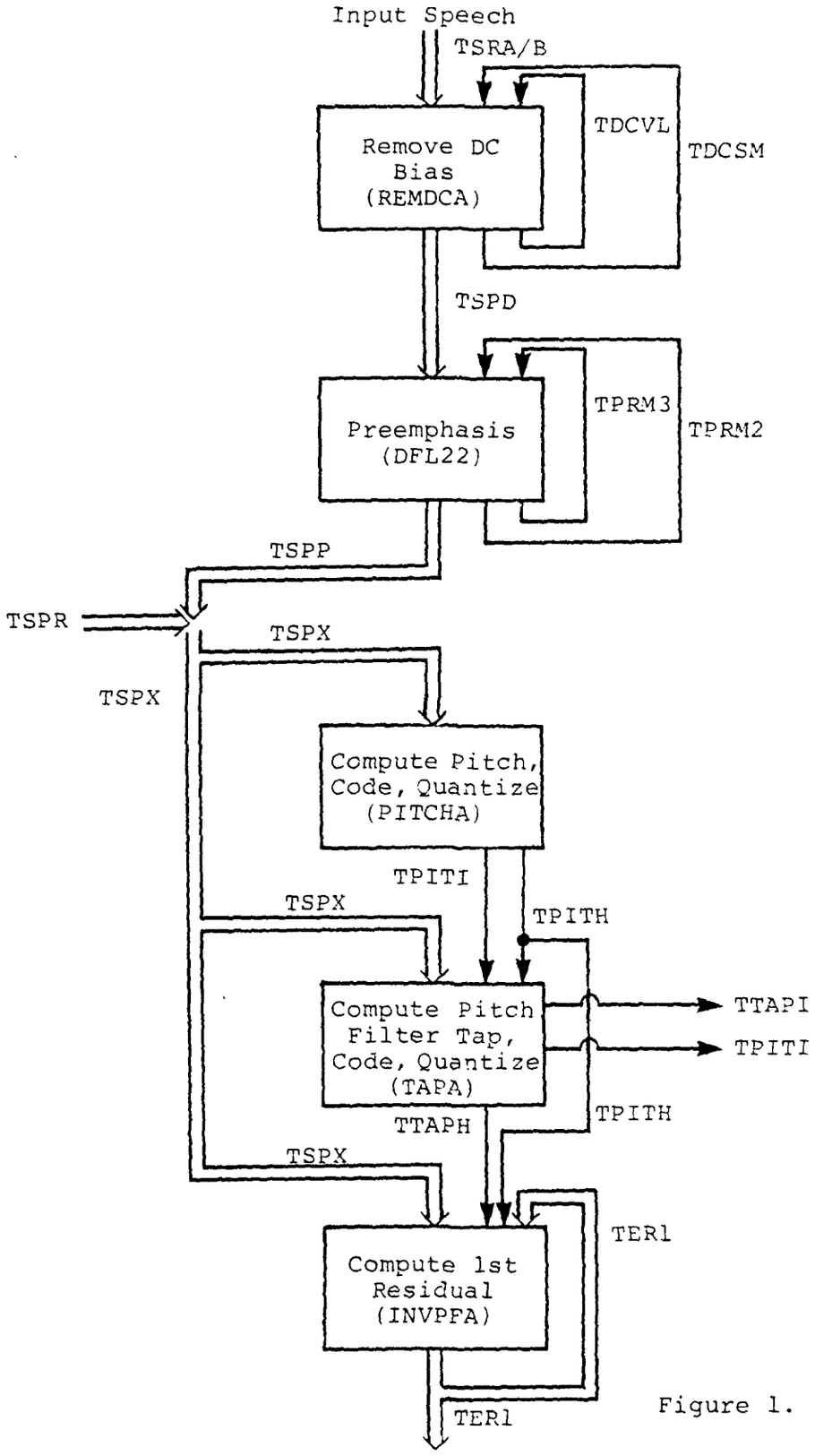


Figure 1. Data flow diagram of APC/SQ analyz e

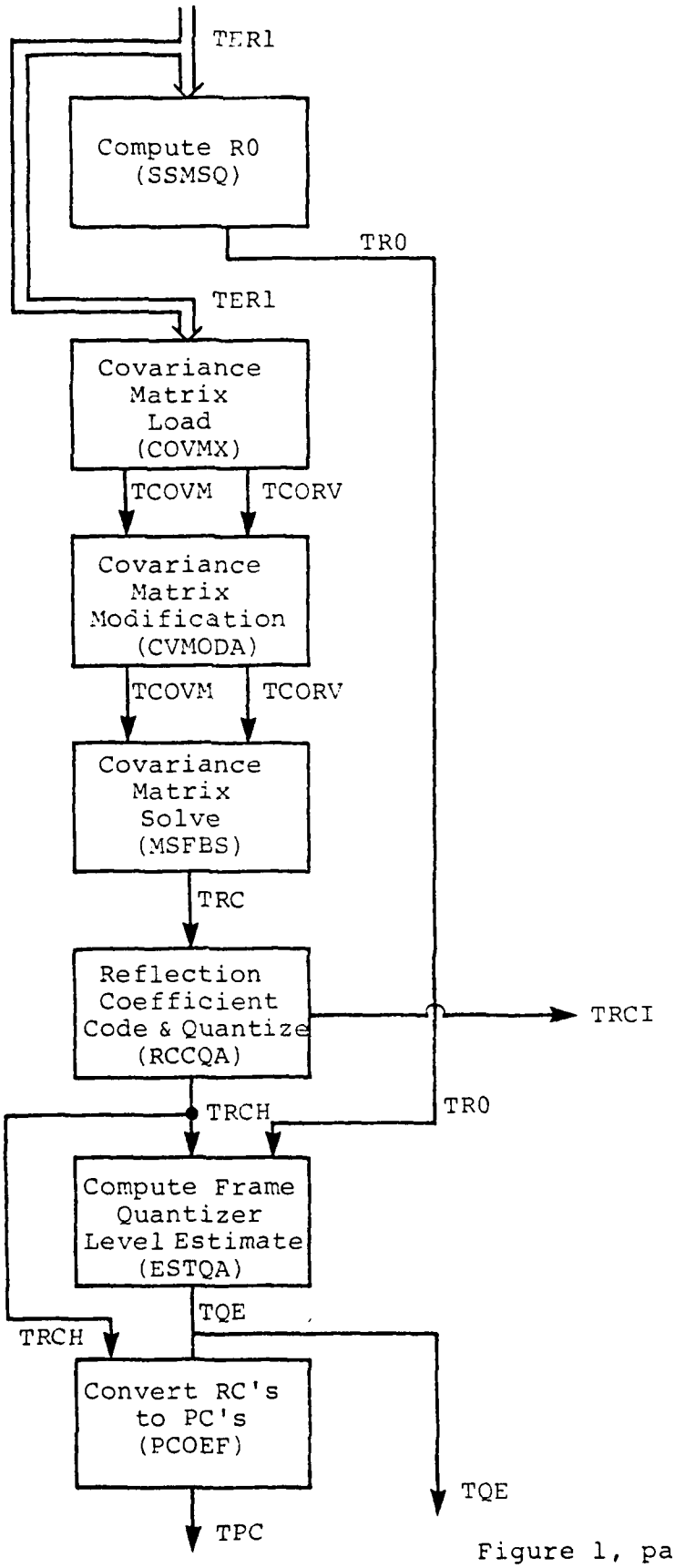


Figure 1, page 2

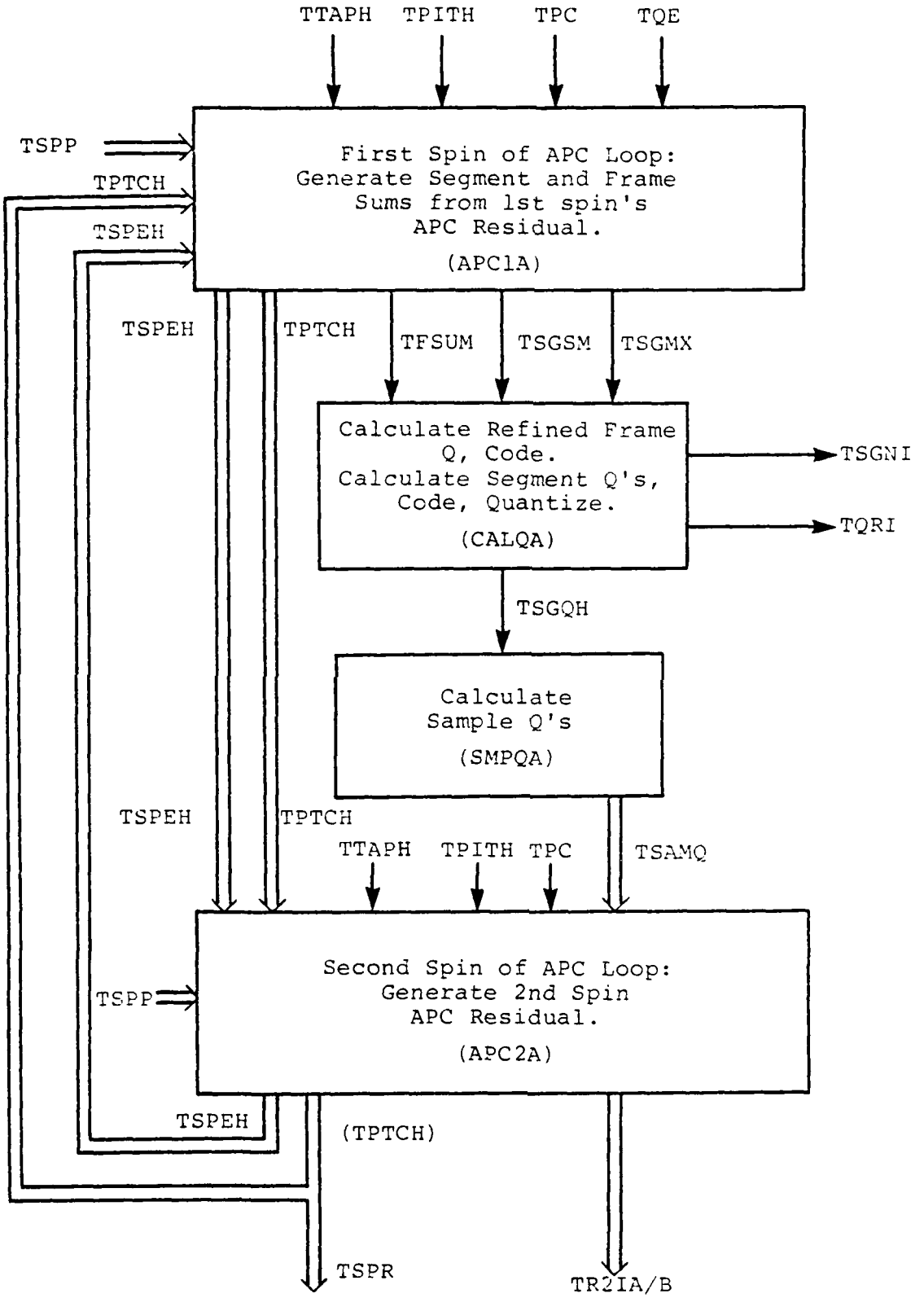


Figure 1, page 3

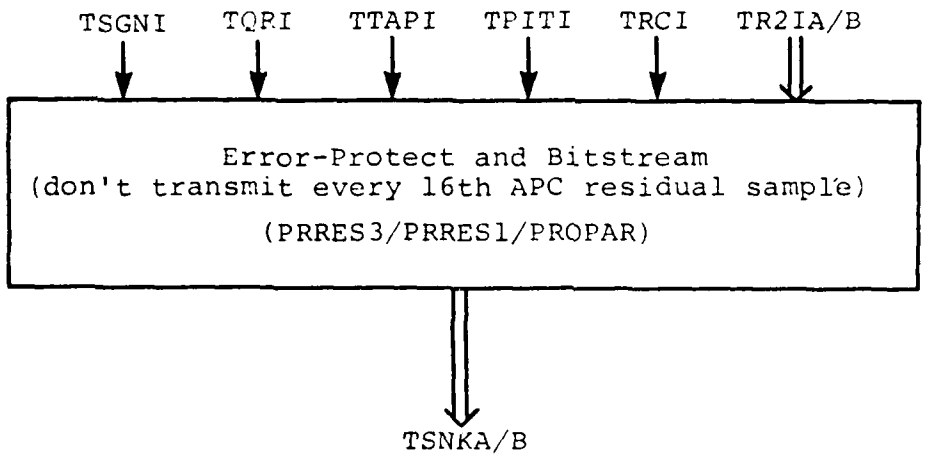
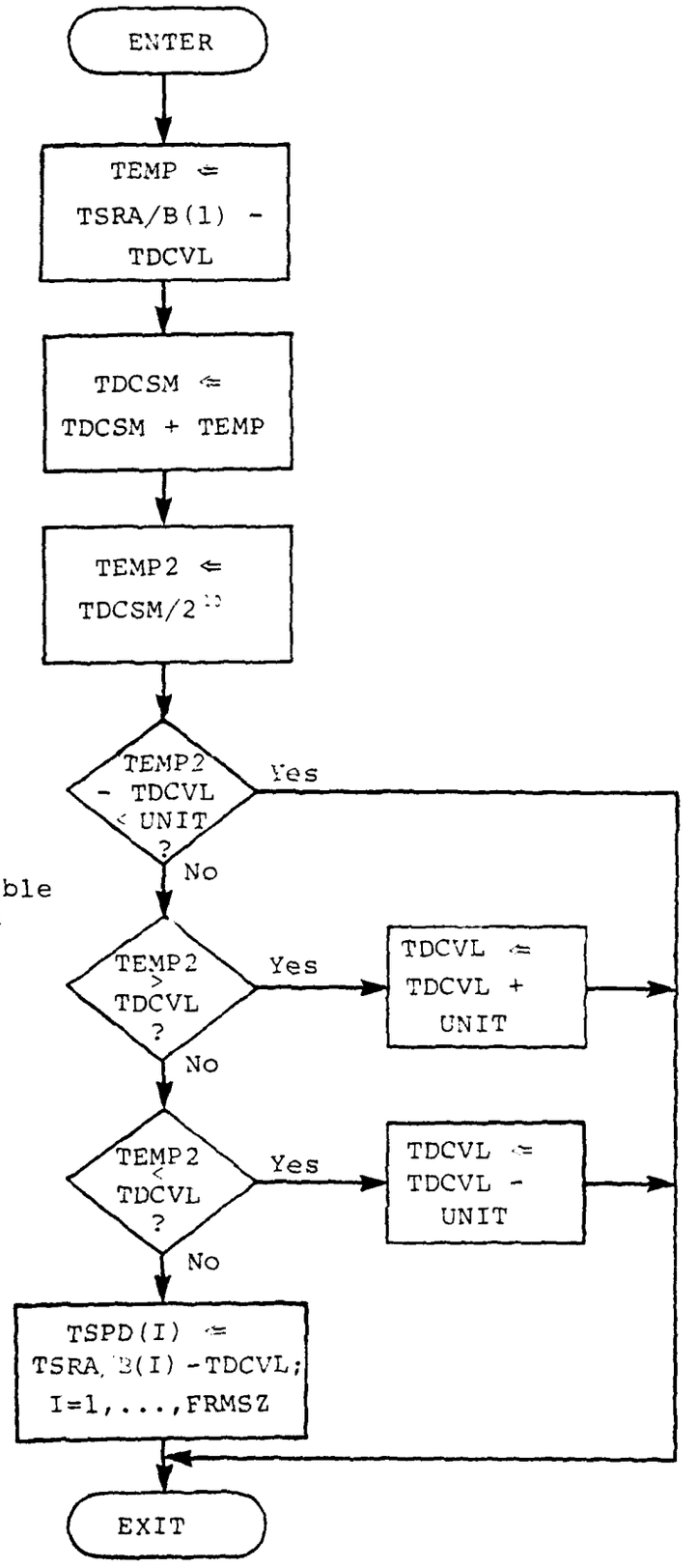


Figure 1, page 4

3.1 Remove D.C. Bias.

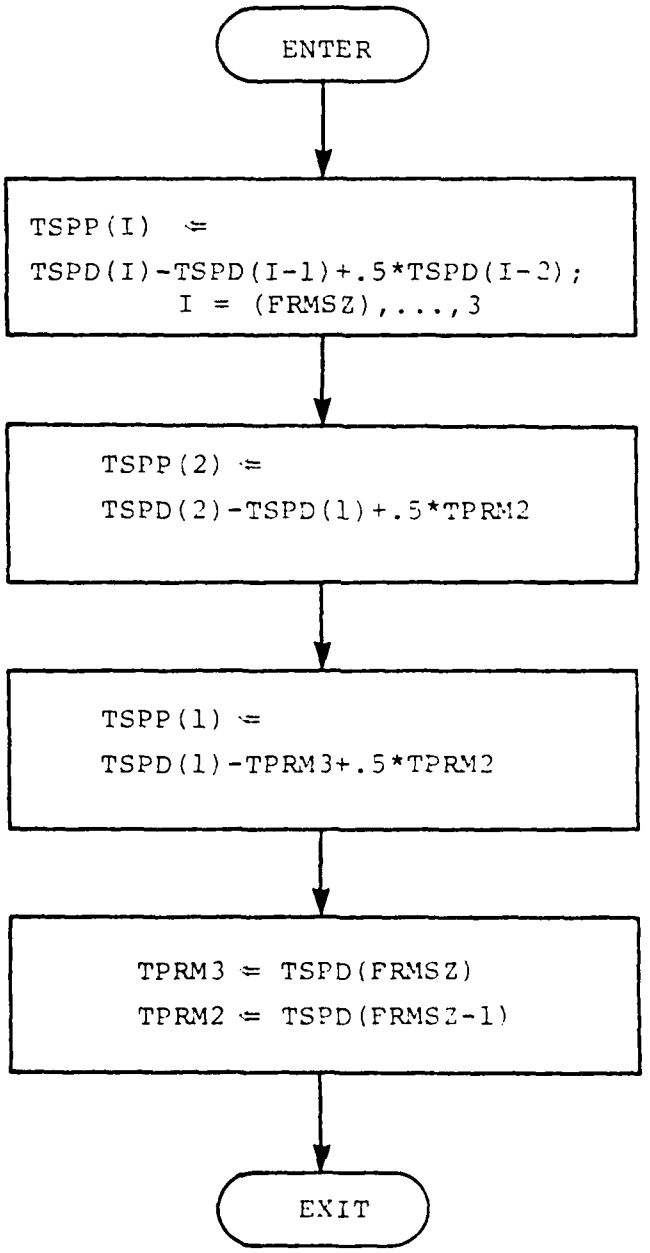


Notes:

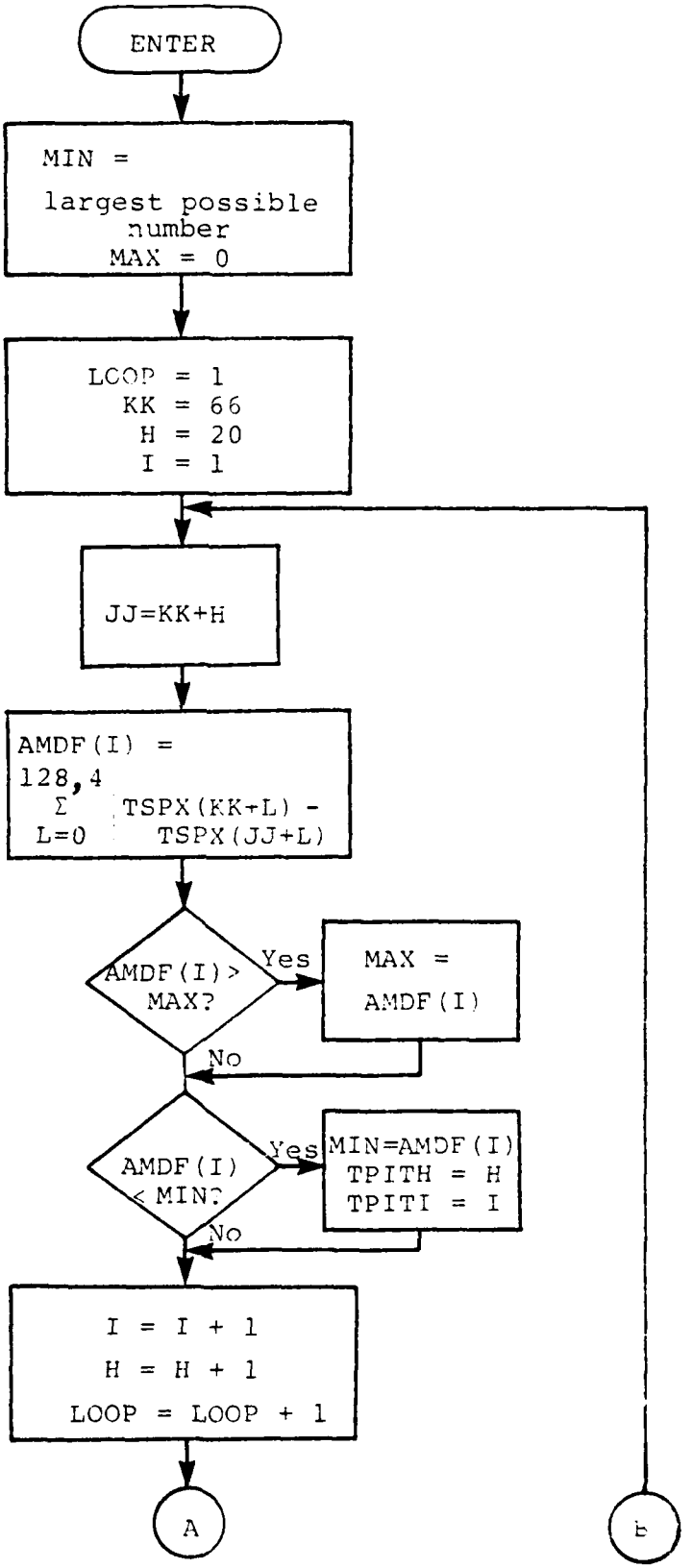
UNIT = ½**11;
i.e., the smallest possible difference between samples. (Samples are 11 bits plus sign, with range -1 to +1.)

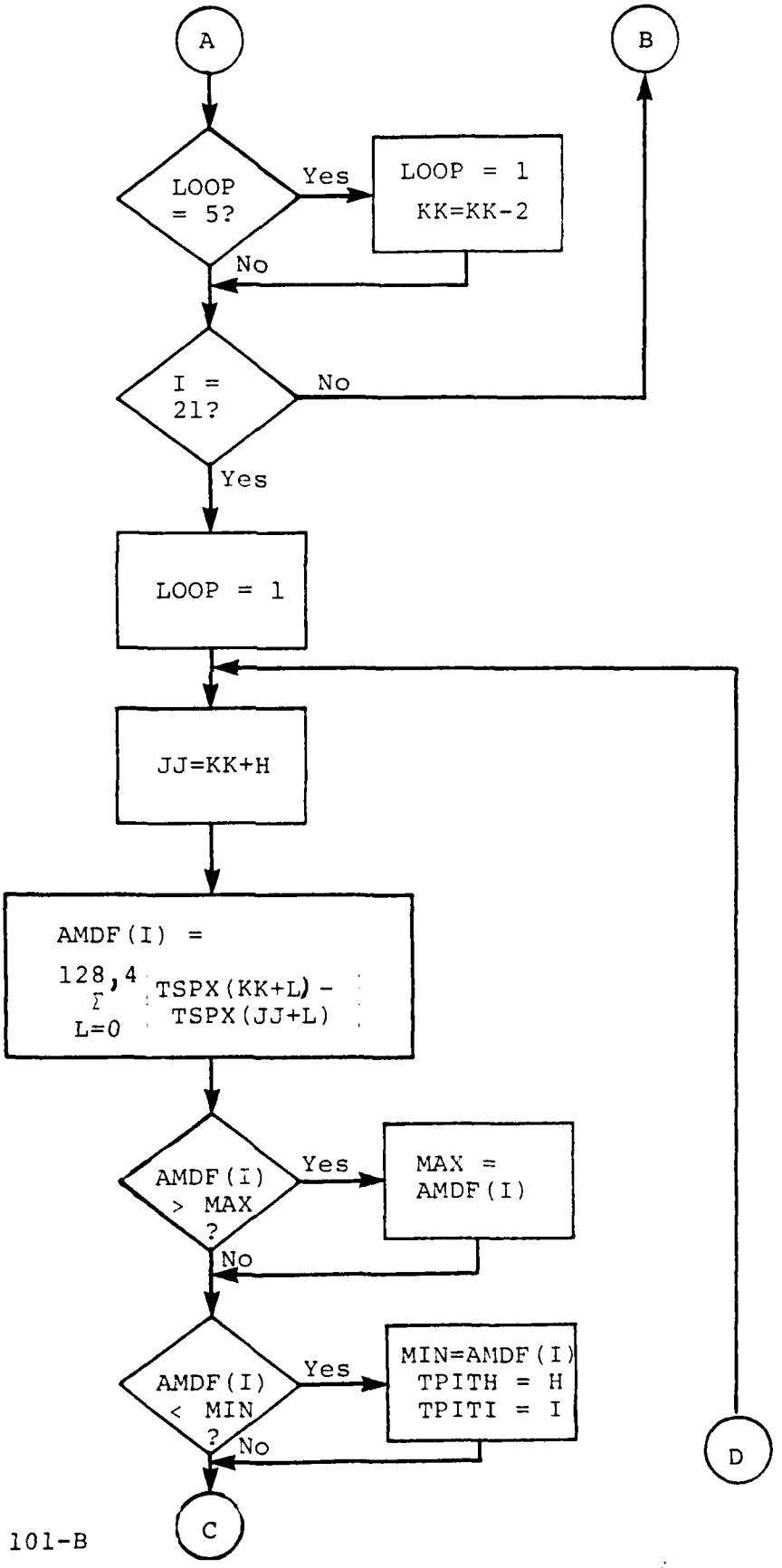
FRMSZ = 192

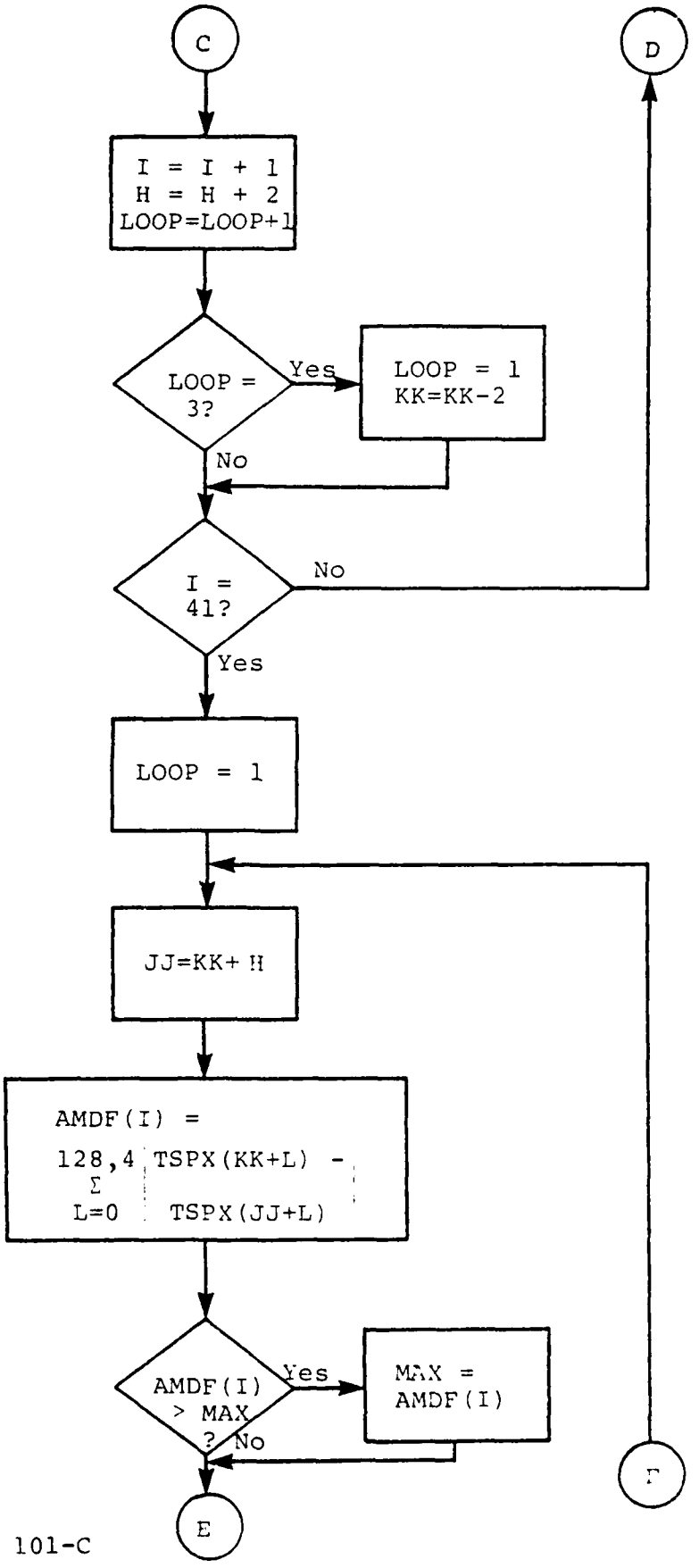
3.2 Preemphasis.

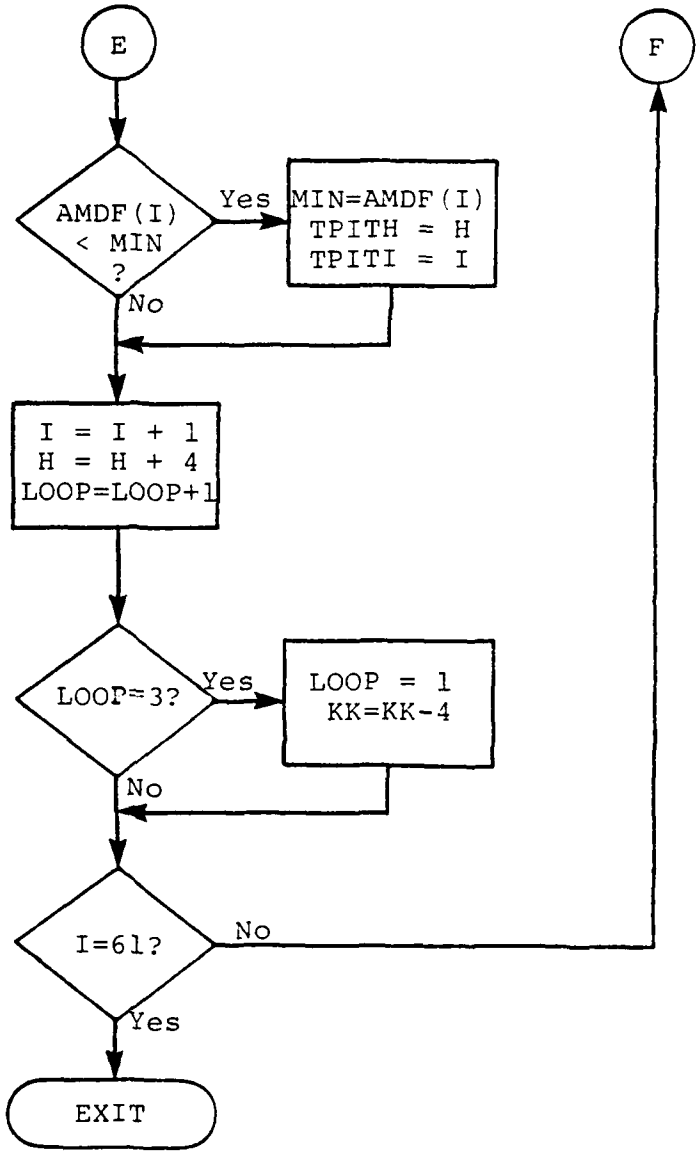


3.3 Compute Pitch, Code, Quantize.

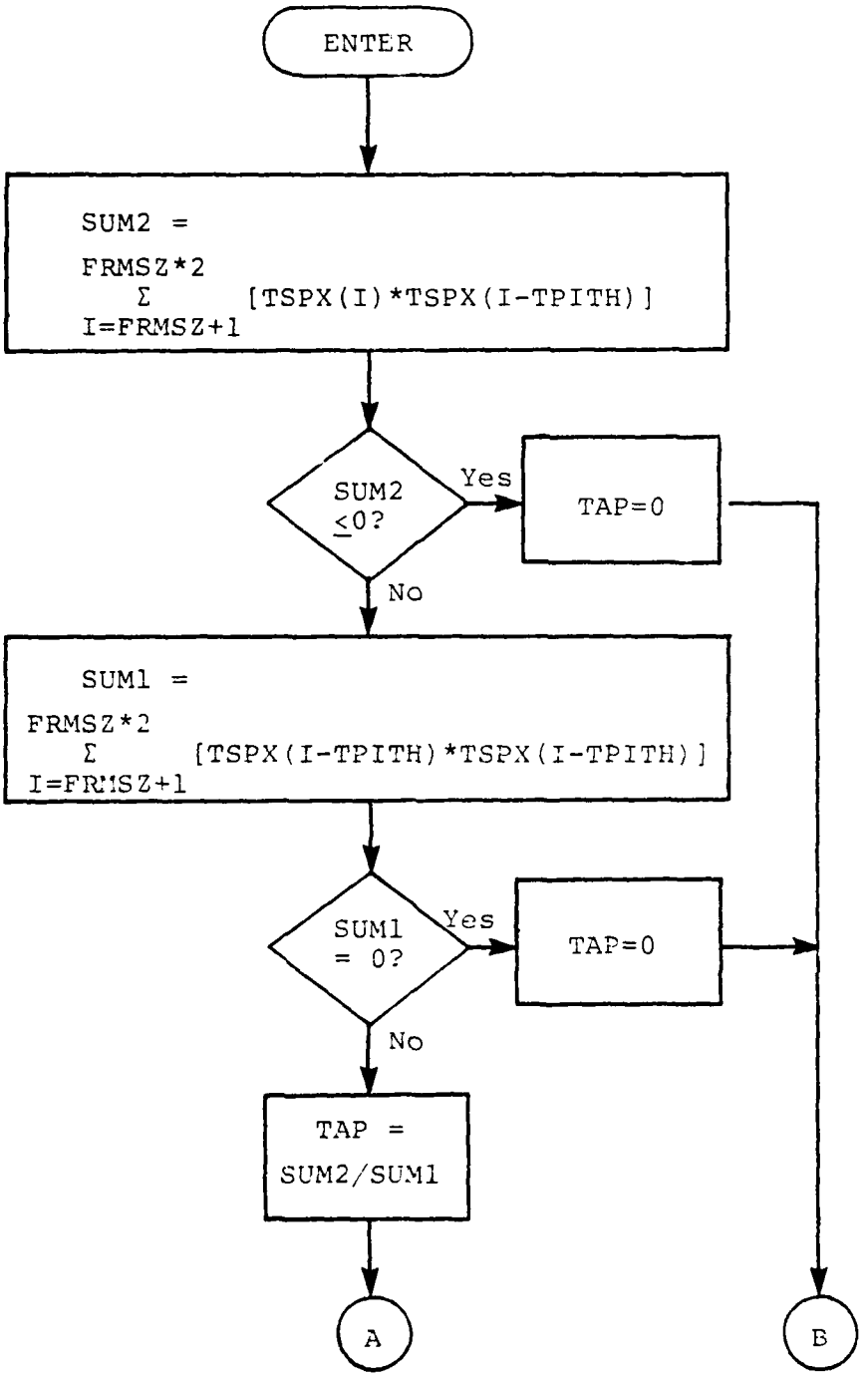


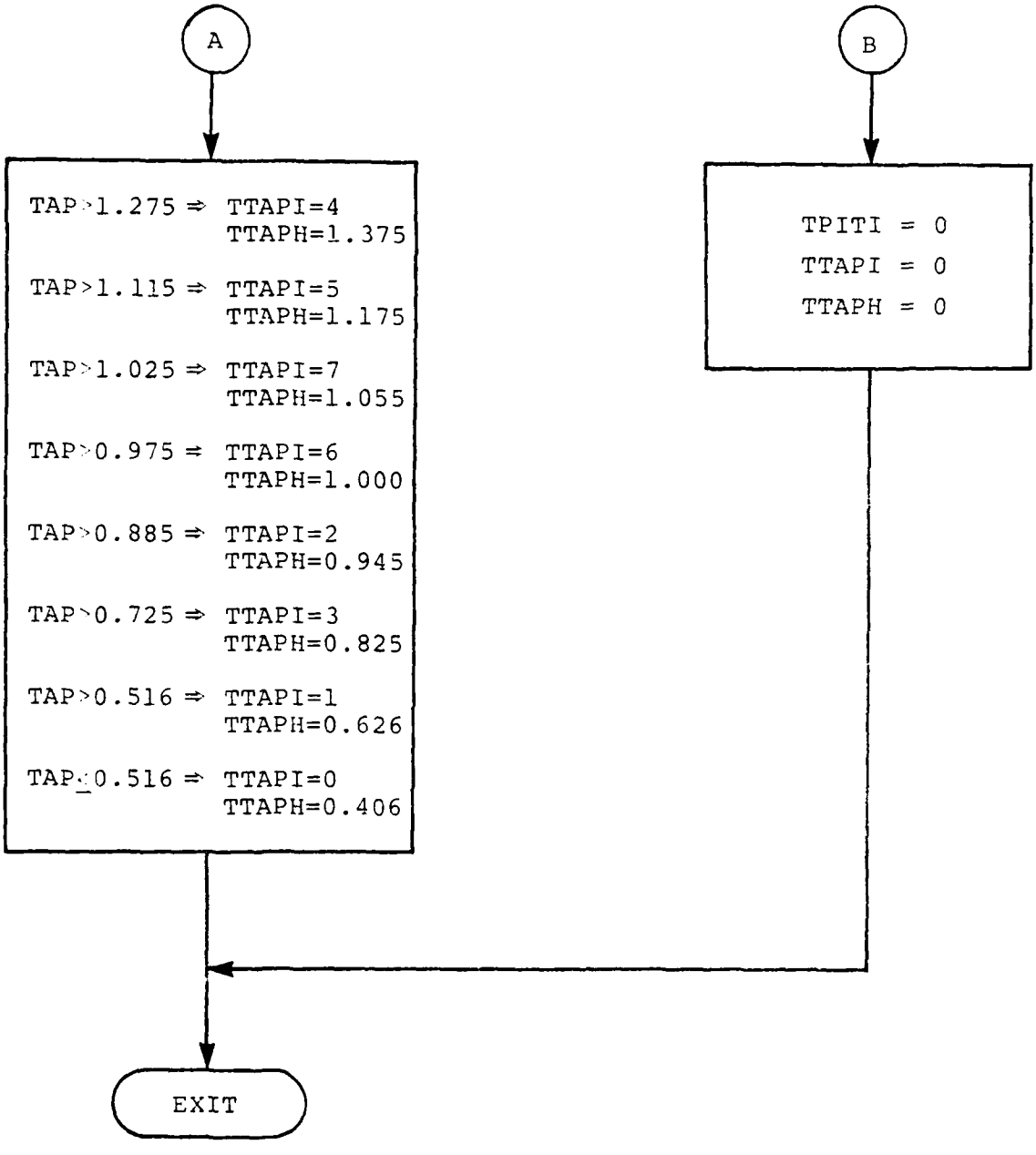

A
B
LOOP = 5?
Yes
LOOP = 1
KK=KK-2
No
I = 21?
No
Yes
LOOP = 1
JJ=KK+H
AMDF(I) =
128,4
Σ
L=0
TSPX(KK+L) - TSPX(JJ+L)
AMDF(I) > MAX ?
Yes
MAX = AMDF(I)
No
AMDF(I) < MIN ?
Yes
MIN=AMDF(I)
TPITH = H
TPITI = I
No
C
D


C
D
I = I + 1
H = H + 2
LOOP=LOOP+1
LOOP = 3?
Yes
LOOP = 1
KK=KK-2
No
I = 41?
No
Yes
LOOP = 1
JJ=KK+H
AMDF(I) =
128,4
Σ
L=0
TSPX(KK+L) -
TSPX(JJ+L)
AMDF(I) > MAX ?
Yes
MAX = AMDF(I)
No
E
F

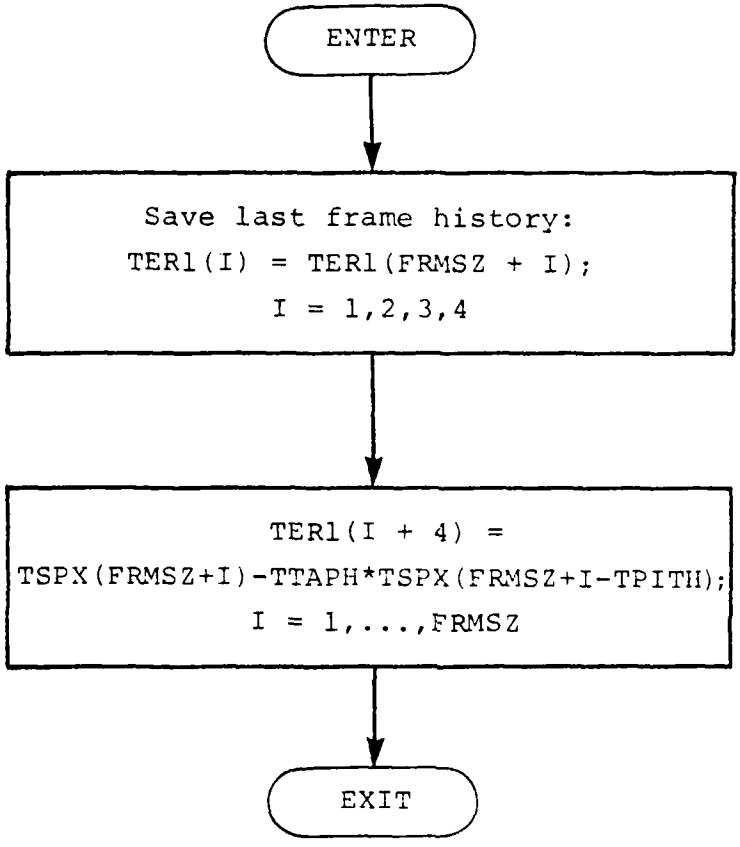

E
F
AMDF(I) < MIN ?
Yes
MIN=AMDF(I)
TPITH = H
TPITI = I
No
I = I + 1
H = H + 4
LOOP=LOOP+1
LOOP=3?
Yes
LOOP = 1
KK=KK-4
No
I=61?
No
Yes
EXIT

3.4 Compute Pitch Filter Tap, Code, Quantize.

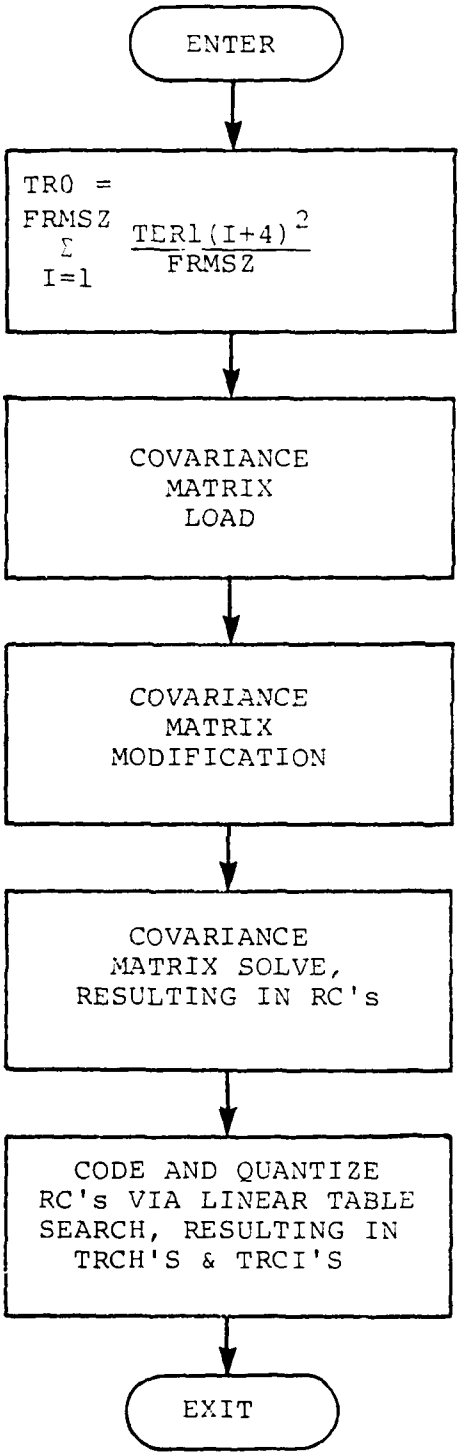



A
B
TAP>1.275 ⇒ TTAPI=4
TTAPH=1.375
TAP>1.115 ⇒ TTAPI=5
TTAPH=1.175
TAP>1.025 ⇒ TTAPI=7
TTAPH=1.055
TAP>0.975 ⇒ TTAPI=6
TTAPH=1.000
TAP>0.885 ⇒ TTAPI=2
TTAPH=0.945
TAP>0.725 ⇒ TTAPI=3
TTAPH=0.825
TAP>0.516 ⇒ TTAPI=1
TTAPH=0.626
TAP≤0.516 ⇒ TTAPI=0
TTAPH=0.406
TPITI = 0
TTAPI = 0
TTAPH = 0
EXIT

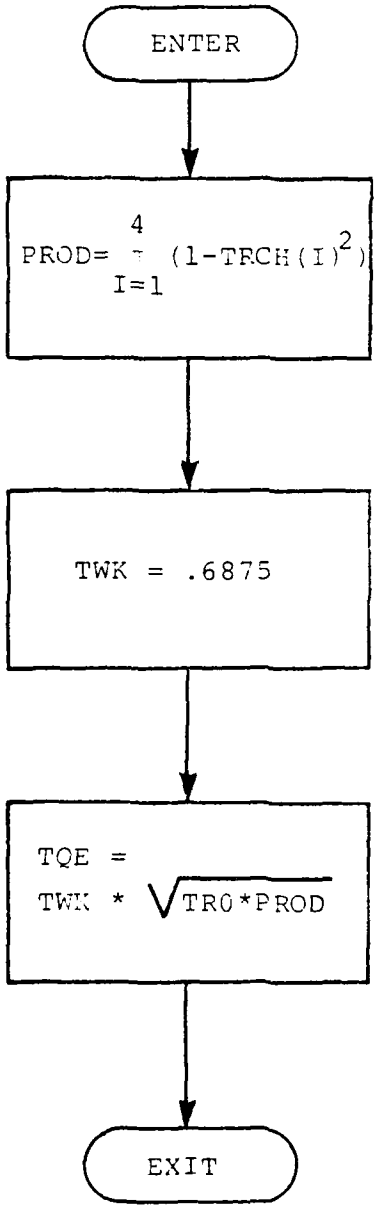
3.5 Compute First Residual.



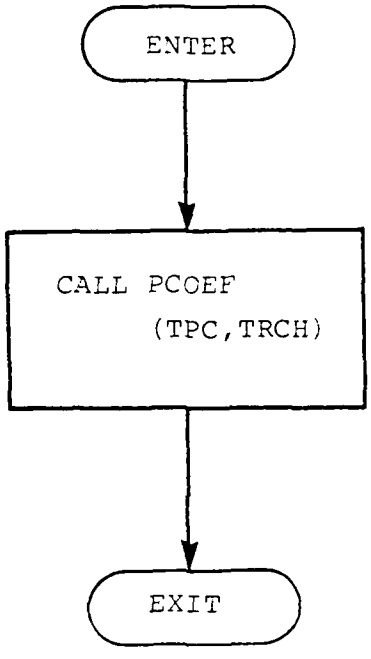
3.6 Compute R0; Compute Reflection Coefficients, Code, Quantize.



3.7 Compute Frame Quantizer Level Estimate.

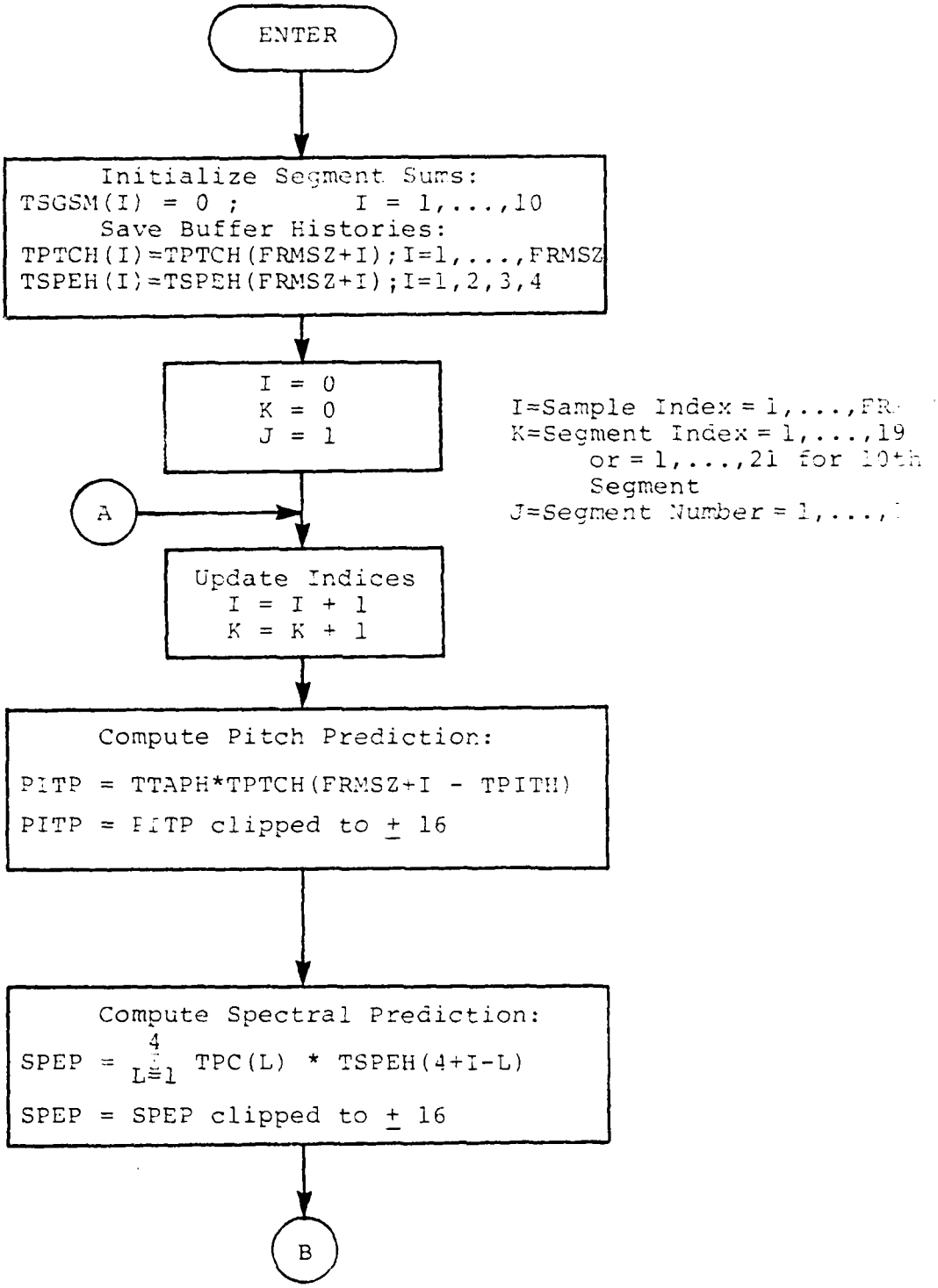


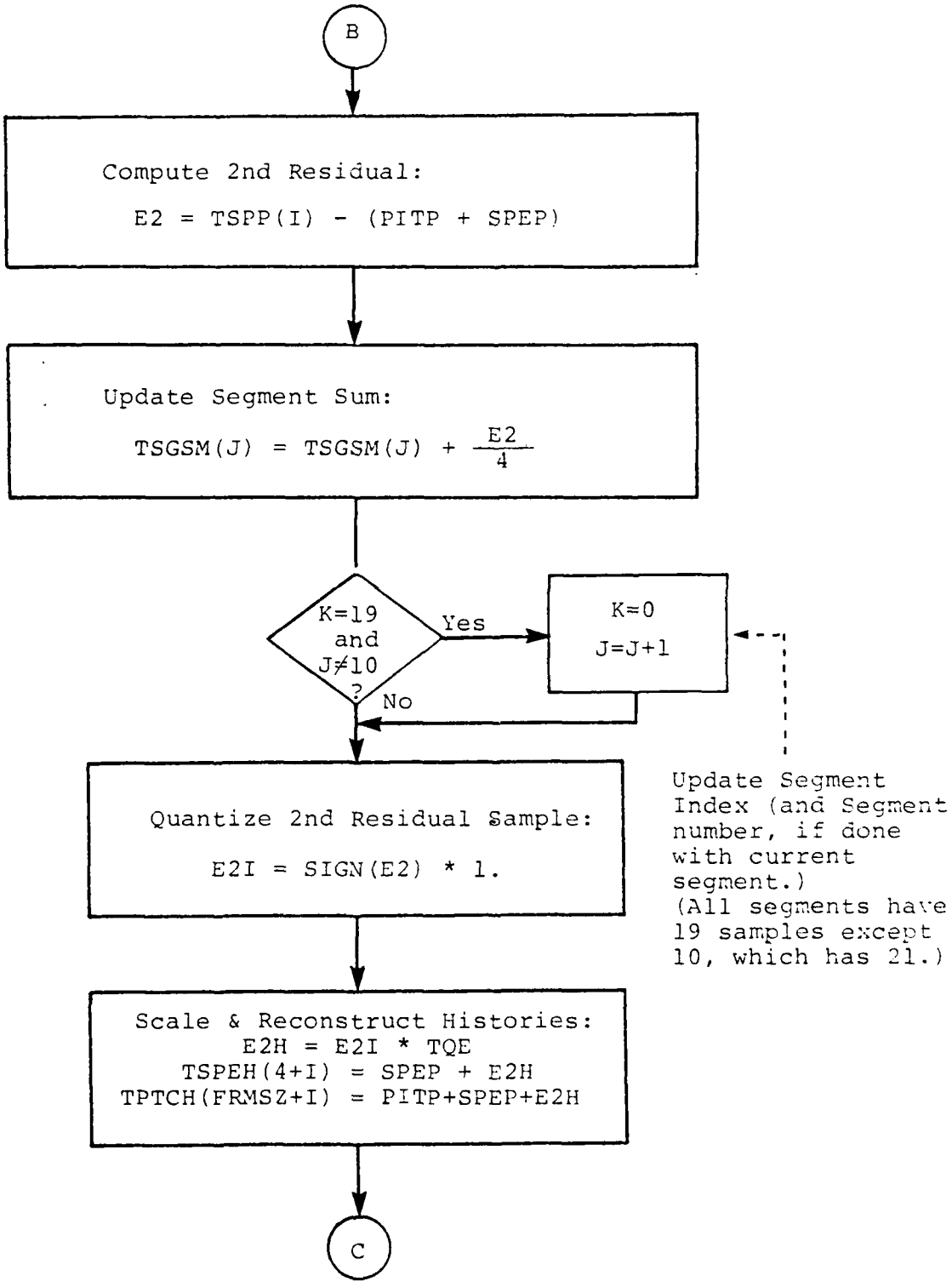
3.8 Convert RC's to PC's.

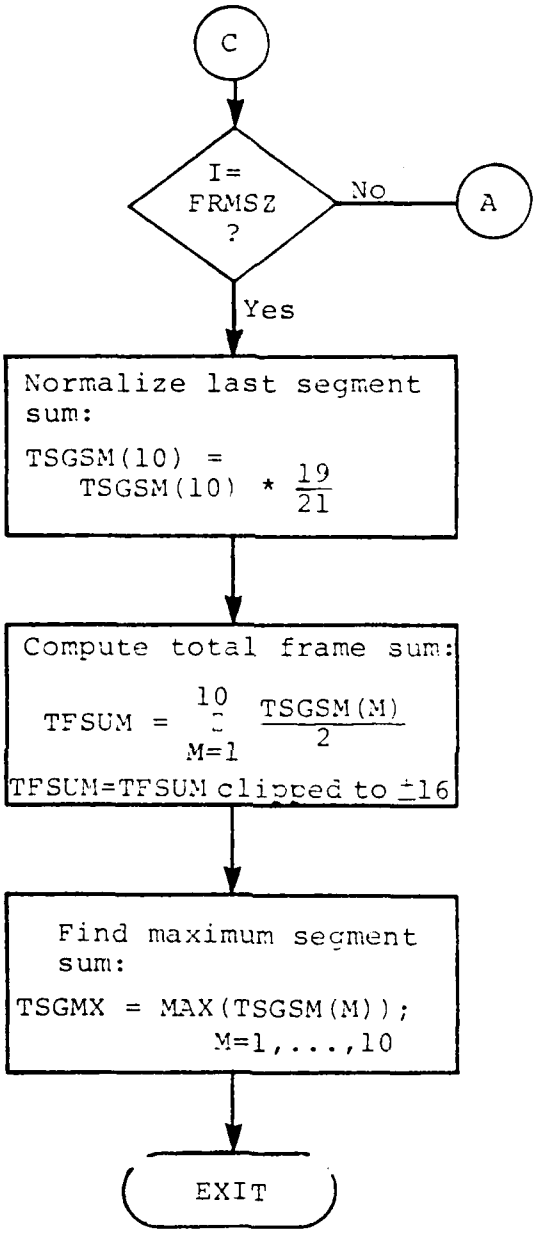


(See Function Description for PCOEF in Appendix B)

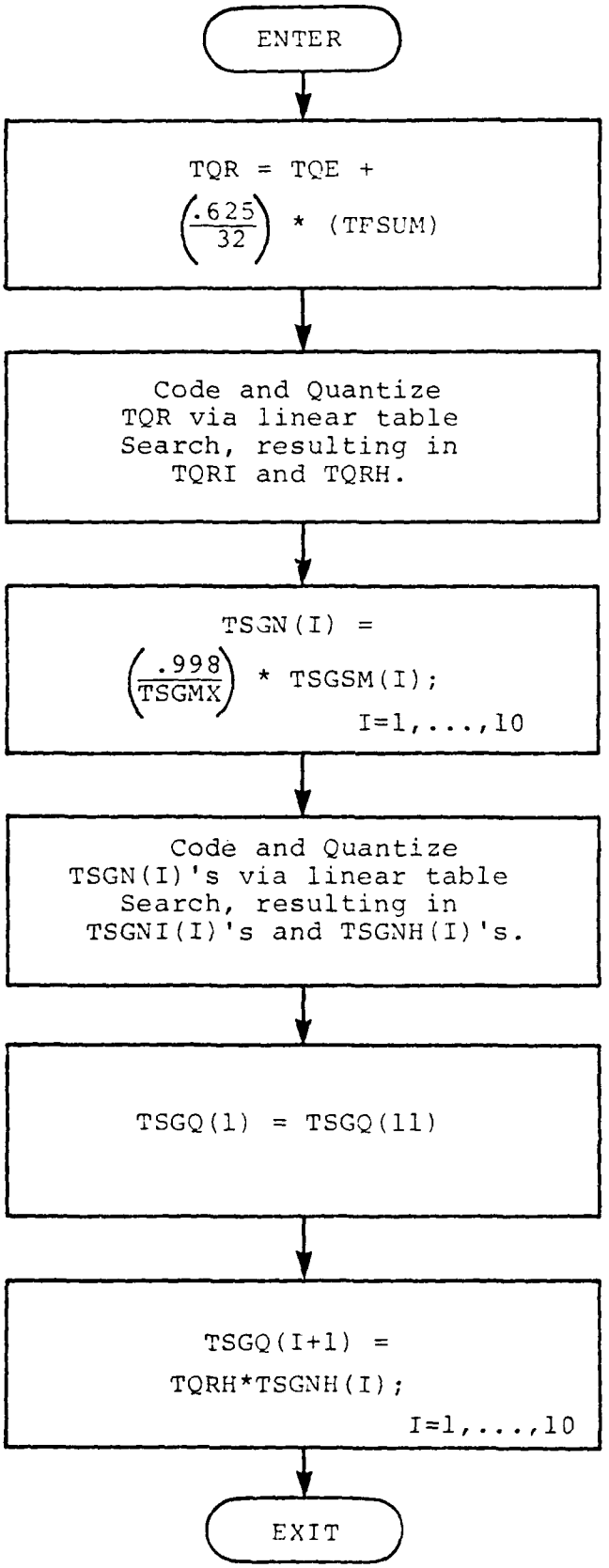
3.9 First Spin of APC Loop: Generate Segment and Frame Sums.



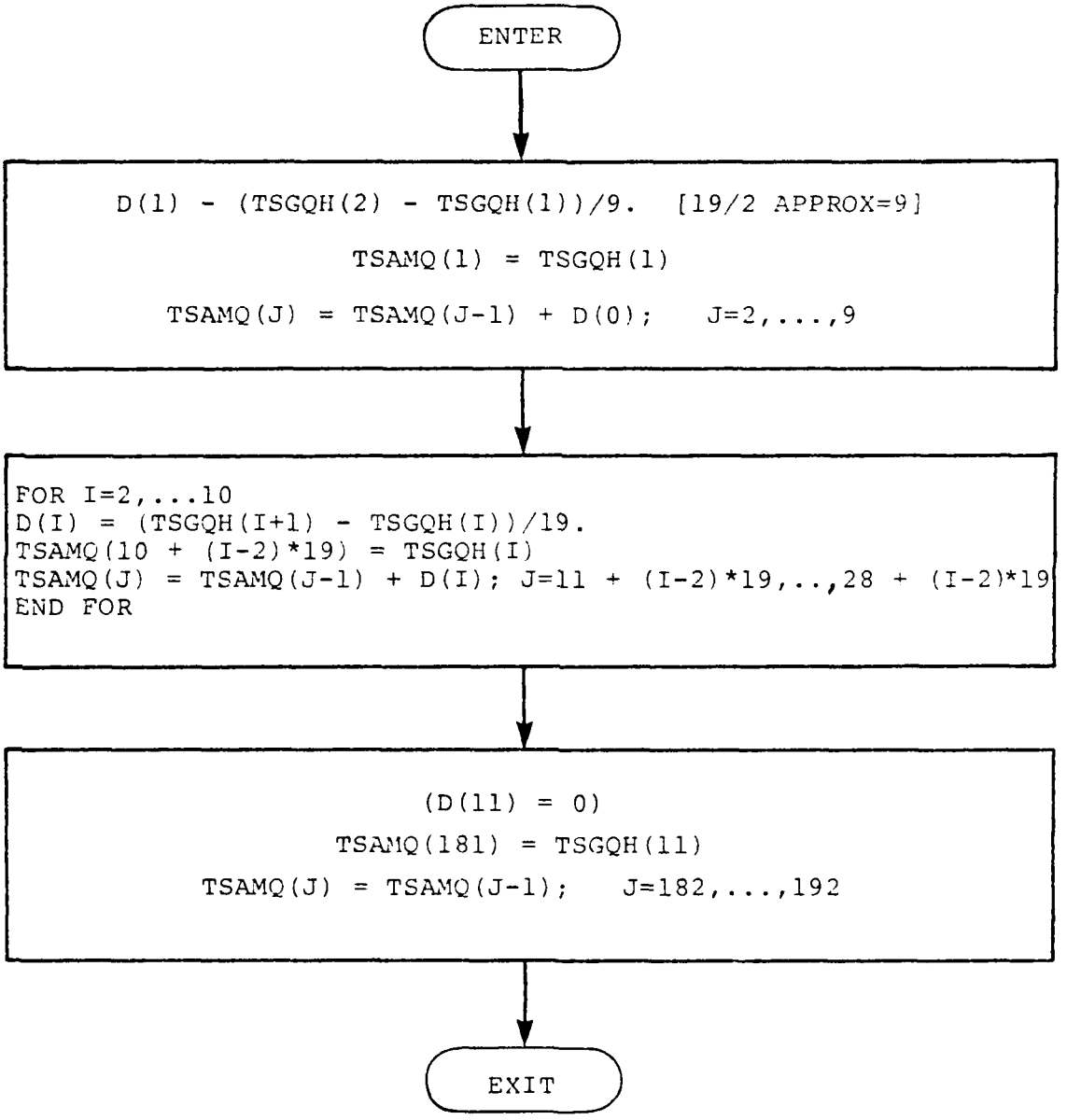

B
Compute 2nd Residual:
E2 = TSPP(I) - (PITP + SPEP)
Update Segment Sum:
TSGSM(J) = TSGSM(J) + E2/4
K=19 and J≠10 ?
Yes
No
K=0
J=J+1
Update Segment Index (and Segment number, if done with current segment.)
(All segments have 19 samples except 10, which has 21.)
Quantize 2nd Residual Sample:
E2I = SIGN(E2) * 1.
Scale & Reconstruct Histories:
E2H = E2I * TQE
TSPEH(4+I) = SPEP + E2H
TPTCH(FRMSZ+I) = PITP+SPEP+E2H
C


C
I= FRMSZ ?
No
A
Yes
Normalize last segment sum:
TSGSM(10) = TSGSM(10) * 19/21
Compute total frame sum:
TFSUM = sum_{M=1}^{10} TSGSM(M)/2
TFSUM=TFSUM clipped to ±16
Find maximum segment sum:
TSGMX = MAX(TSGSM(M)); M=1,...,10
EXIT

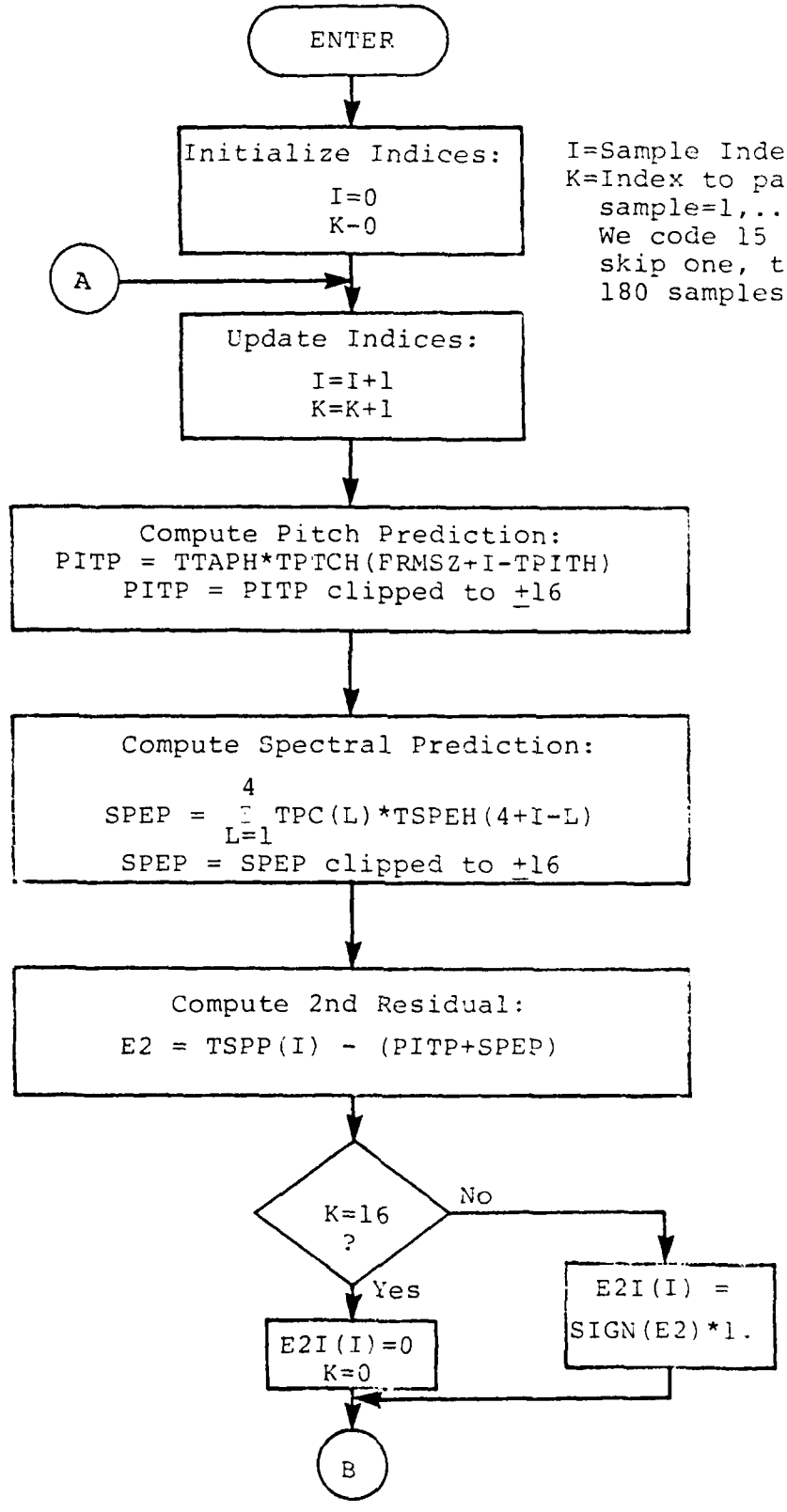
3.10. Calculate Refined Frame Q, Code, Quantize.
Calculate Segment Q's, Code, Quantize.

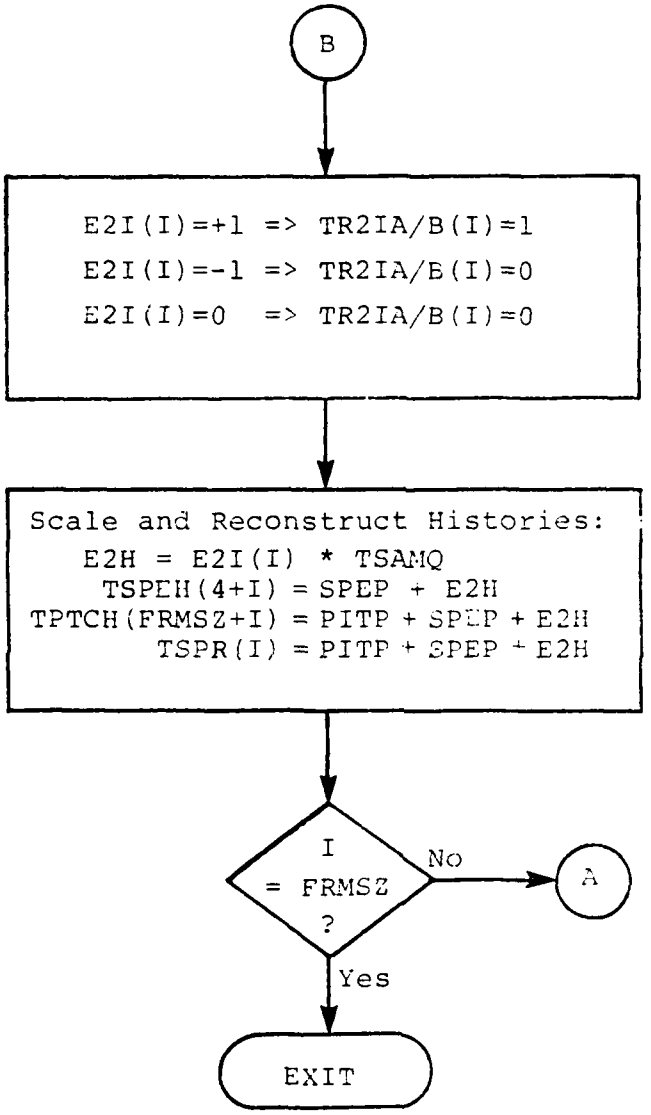


3.11 Calculate Sample Q's.



3.12 Second Spin of APC Loop: Generate APC Residual.





Output coded APC Residual Samples.

3.13 Error-Protect and Bitstream

See Function Descriptions for PRRES3, PRRES1, and PROPAR in Appendix B for a description of this module.

4. Synthesizer

A data flow diagram of the synthesizer is given in Fig. 2. In this section, we provide a control flowchart for each of the various synthesizer components. Throughout this section, data buffer/scalar names and processing function names are those used in the MAP-300 implementation of APC/SQ. Arrays are subscripted starting at 1.

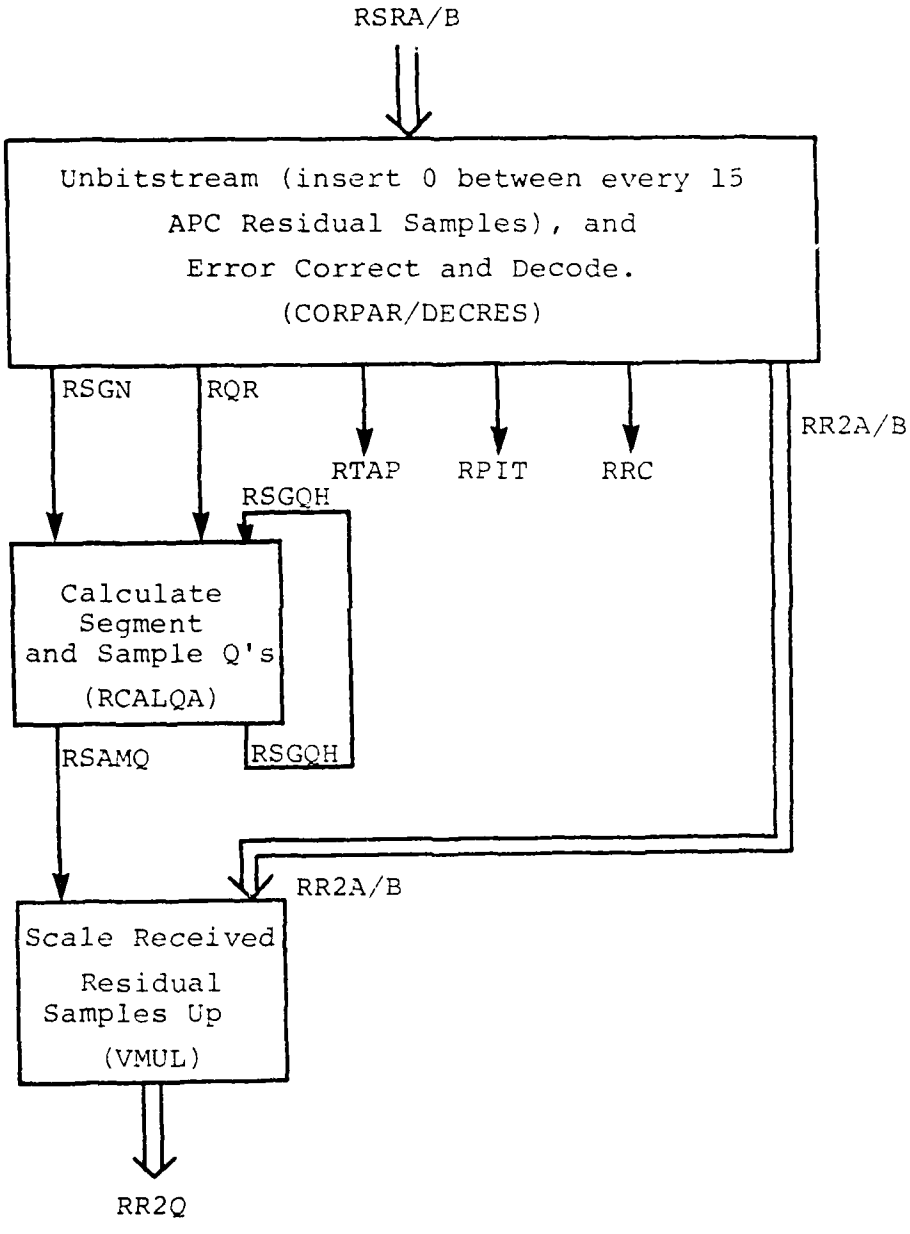
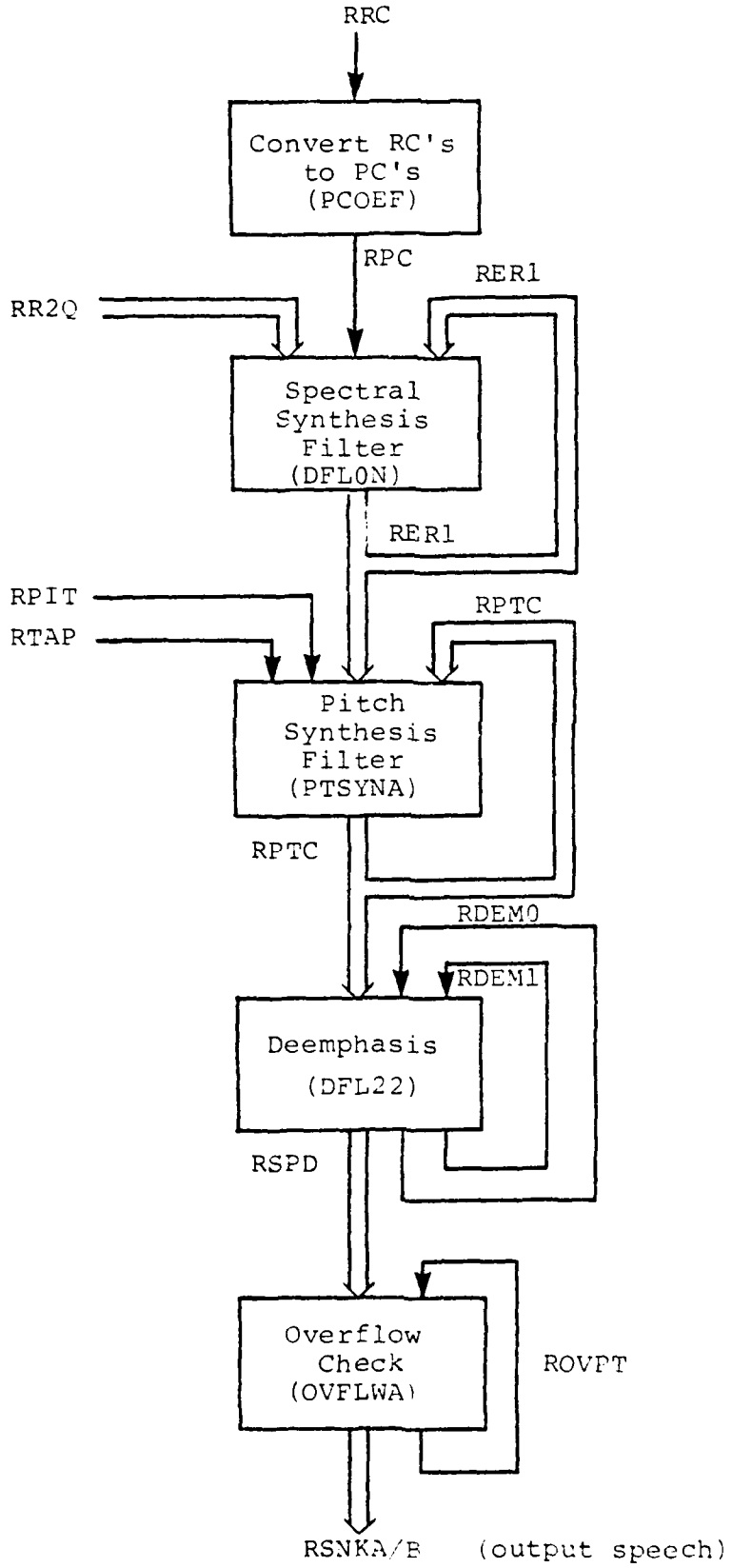


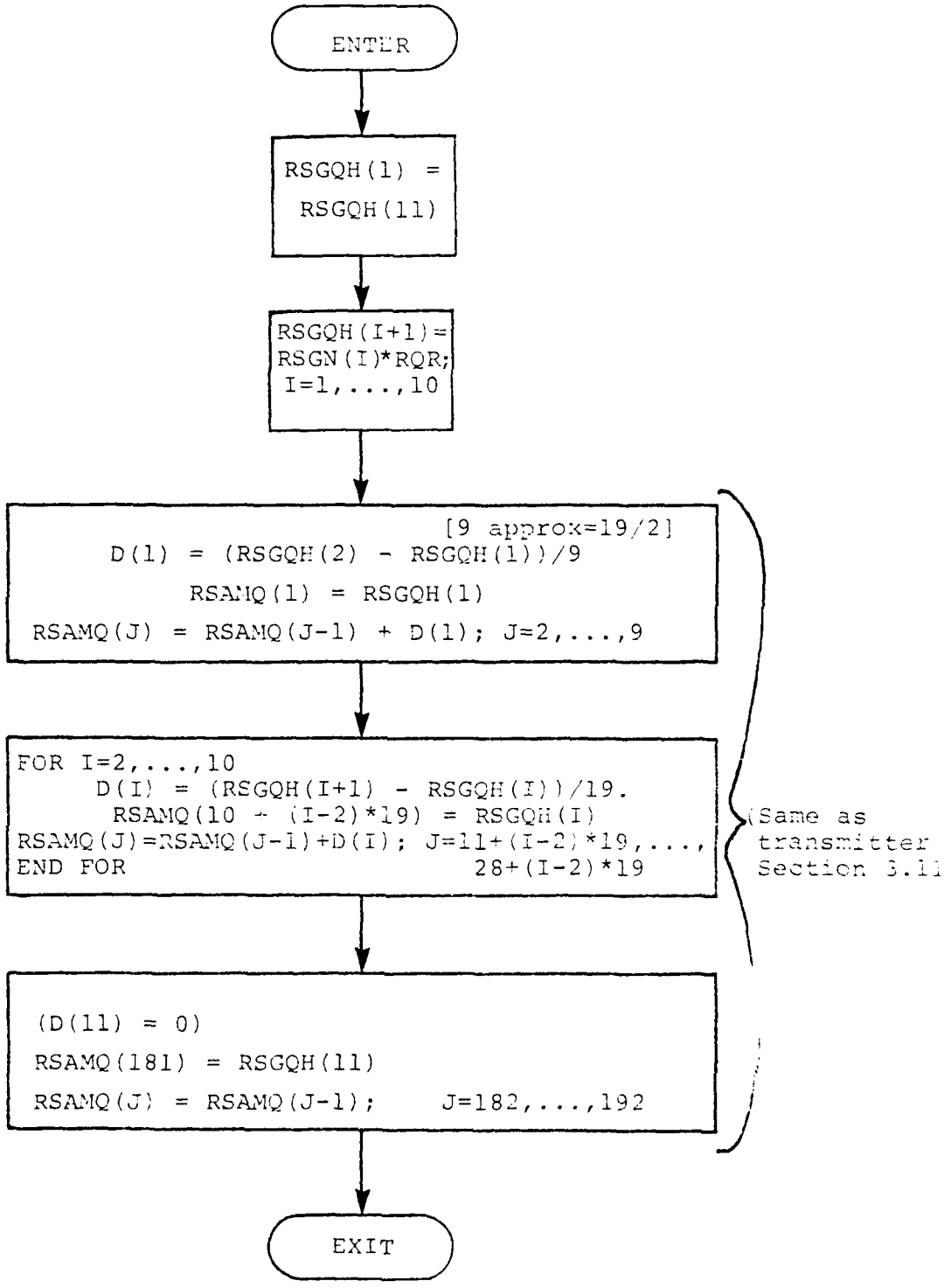
Figure 2. Data flow diagram of APC/SQ synthesizer.


RRC
Convert RC's to PC's (PCOEF)
RPC
RER1
RR2Q
Spectral Synthesis Filter (DFL0N)
RER1
RPIT
RTAP
RPTC
Pitch Synthesis Filter (PTSYNA)
RPTC
RDEM0
RDEM1
Deemphasis (DFL22)
RSPD
Overflow Check (OVFLWA)
ROVFT
RSNKA/B (output speech)

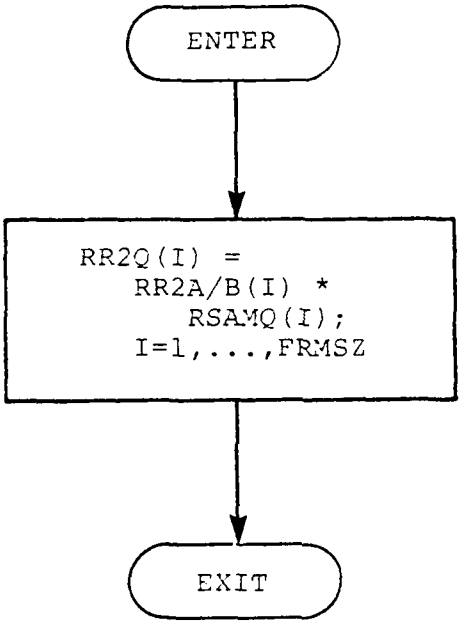
4.1 Unbitstream, Error-Correct and Decode

See Function Descriptions for CORPAR and DECRES in Appendix B for a description of this module.

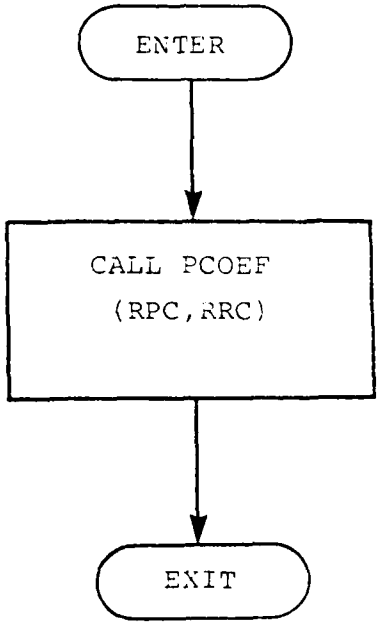
4.2 Calculate Segment and Sample Q's.



4.3 Scale Received Residual Samples Up By Sample Q's.

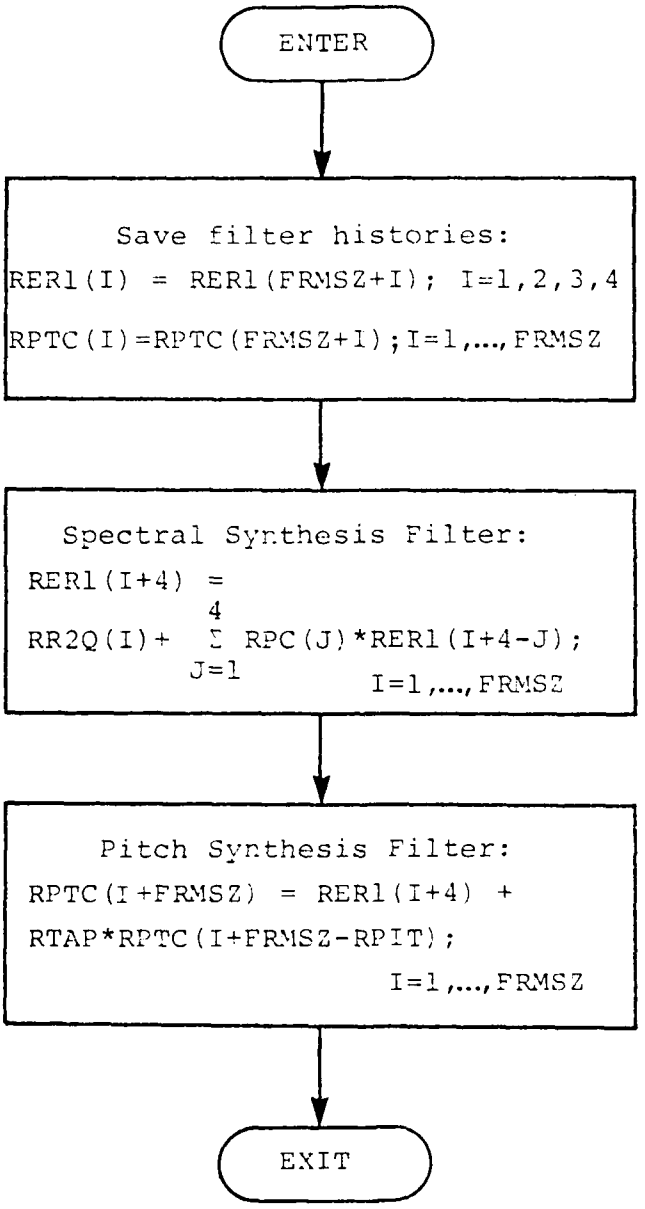


4.4 Convert RC's to PC's.

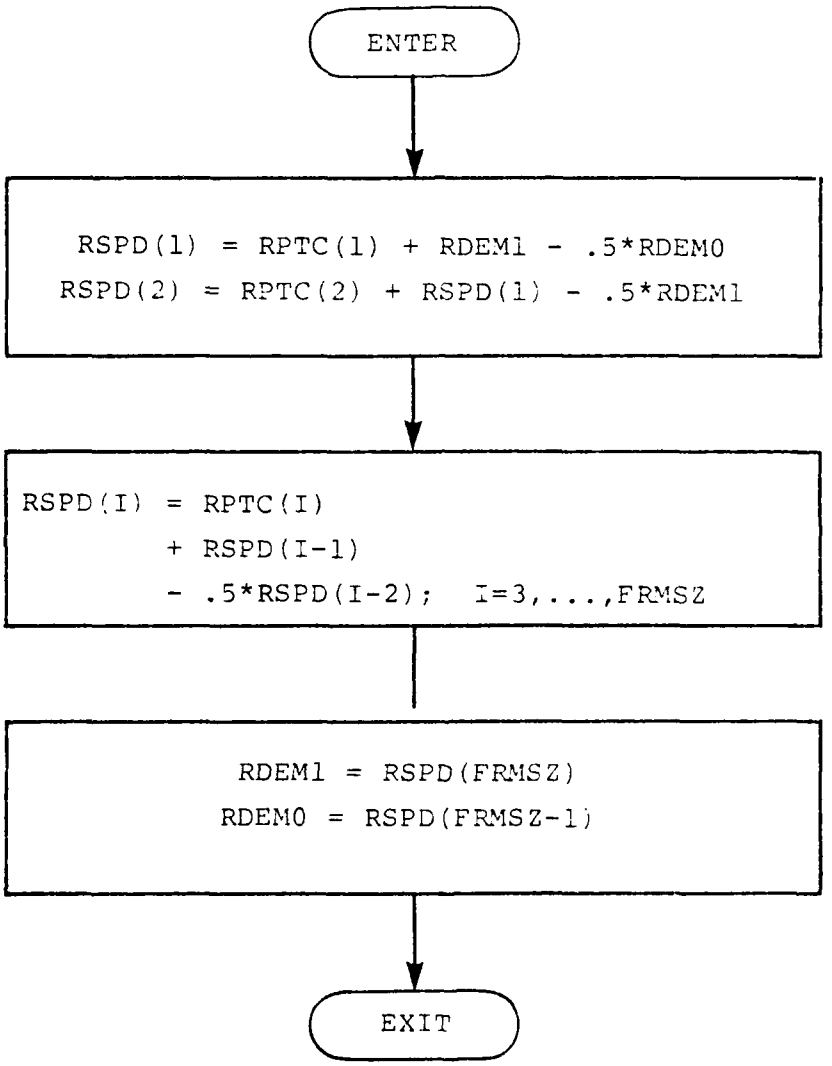


(See Function Description for PCOEF in Appendix B)

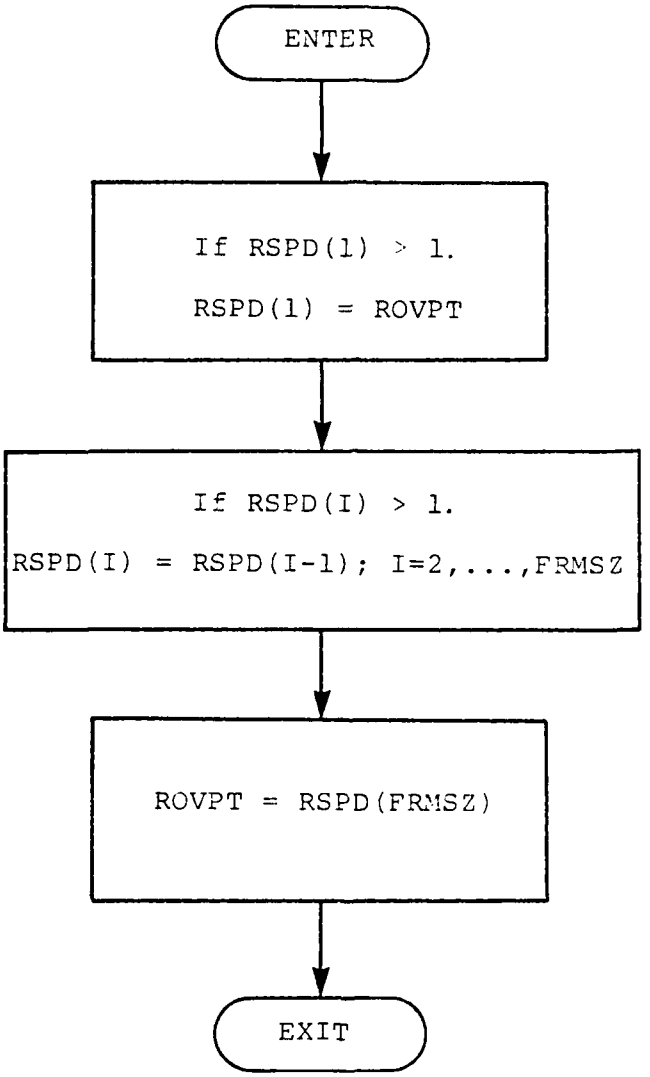
4.5 Spectral Synthesis Filter, Pitch Synthesis Filter.



4.5 Deemphasis



4.7 Overflow Check.



APPENDIX B
BBN-WRITTEN MAP-300 FUNCTIONS FOR APC/SQ

This Appendix contains a complete user-level description for each MAP-300 SNAP-II function written by BBN and used in the APC/SQ speech coder implementation. The descriptions are ordered alphabetically by function name.

Functions that were used in the speech coder implementation and are not described here were supplied by CSPI as part of the SNAP-II Software System (See Sections 2.5.1.1 and 2.5.2.1).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: ADINT
NAME EXPANSION: Simulate A/D interrupt

FCB #: 124
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: ADAMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

This SNAP callable function calls the ADAMINT A/D interrupt Service module in exactly the same manner as it is used to respond to A/D interrupts, so it may be used to simulate such an interrupt.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: ADPEAK(U)
NAME EXPANSION: Monitor peak signal from the A/D

FCB #: 188
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: ADPKU
APS MODULE NAME: ADPKA
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer U	1-63	Real	Input

NUMBER OF OUTPUT SAMPLES: 1, in integer scalar TADPK
NOTES: -

FUNCTION DESCRIPTION:

Maintains a running peak absolute value of the data in the input boffers given to it on successive calls. The running peak is maintained as an integer value (to facilitate monitoring the peak level) in the integer scalar TADPK. Since SNAP-II standard binding does not support binding an integer scalar to an array function, the address of TADPK is "hard bound" in the APS module source code.

The input buffer is assumed to contain signals in the range -1 to +1 from a 12-bit A/D converter. The integer value maintained in TADPK falls in the range 0-2047.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: AHISTA (S,T,U,V,W,X,Y,Z)
NAME EXPANSION: Set up analysis frame histories.

FCB #: 132
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: AHUS
APS MODULE NAME: AHSS
CSPU MODULE NAME: SBMSAH

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer S	1-63	Real	Output: First segment Q
Buffer T	1-63	Real	Input: Last Segment Q
Buffer U	1-63	Real	Output: 1st residual history, current frame
Buffer V	1-63	Real	Input: 1st residual: last 4 elements of current frame
Buffer W	1-63	Real	Output: APC pitch filter history: last, current frames
Buffer X	1-63	Real	Input: APC pitch filter history: current frame
Buffer Y	1-63	Real	Output: APC spectral filter history: history, current frame

NUMBER OF OUTPUT SAMPLES: TBS + VBS + XBS + ZBS
NOTES: Requires special CSPU binding to accomodate 6 buffers.

FUNCTION DESCRIPTION:

S(0) - S(TBS-1) <= T(0) - T(TBS-1)
U(0) - U(VBS-1) <= V(0) - V(VBS-1)
W(0) - W(XBS-1) <= X(0) - X(XBS-1)
Y(0) - Y(ZBS-1) <= Z(0) - Z(ZBS-1)

PARAMETER DEFINITIONS:(con't)

Buffer Z	1-63	Real	Input: APC spectral filter history: last 4 elements of current frame

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: APC1A(T,U,V,W,X,A,B,C)
NAME EXPANSION: APC 1st Spin, Plus Segment Sum Accumulations

FCB #: 177
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: A1US
APS MODULE NAME: A1SS
CSPU MODULE NAME: SBMSA1

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer T	1-63	Real(4)	Input: Spectral Predictor Coefficients
Buffer U	1-63	Real (192)	Input: Current frame preemphasized input speech
Buffer V	1-63	Real (384)	Input/Output: Pitch prediction Buffer
Buffer W	1-63	Real (196)	Input/Output: Spectral prediction buffer
Buffer X	1-63	Real (10)	Output: Segment sums
Real Scalar A	0-191	Real	Input: 1st of 3 consec. scalars Quant. Tap, Quant Pitch, Frame 0 Est.
Real Scalar B	0-191	Real	Output: Frame Sum

NUMBER OF OUTPUT SAMPLES: 12
NOTES: This routine uses (SA+1) to modify its APS code. SA is first of 3 contiguous scalars (grouped due to limit of 8 function parameters). Special binding is required to handle parameter mixture.
FUNCTION DESCRIPTION:

Generate APC residual estimate using single quantizer level per frame (SA+2). Use this residual estimate to generate synthesized speech, from which segment and frame sums are computed. See Section 3.9 of the Algorithm Specification (Appendix A of this report) for complete description. Note that, of 10 segments per 192 sample frame, segments 1-9 are 19 samples long, and segment 10 is 21 samples long.

PARAMETER DEFINITIONS: (con't)

Real Scalar C	0-191	Real	Output: Maximum Segment Sum

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: APC2A(T,U,V,W,X,Y,Z,A)
NAME EXPANSION: APC 2nd Spin: Generate APC residual

FCB #: 18C
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: A2US
APS MODULE NAME: A2SS
CSPU MODULE NAME: SBMSA2

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer T	1-63	Real (4)	Input: Spectral predictor coeffs
Buffer U	1-63	Real (192)	Input: Current frame preemphasized input speech
Buffer V	1-63	Real (384)	Input/Output: Pitch prediction buffer
Buffer W	1-63	Real (196)	Input/Output: Spectral prediction buffer
Buffer X	1-63	Real (192)	Input: Sample Q's
Buffer Y	1-63	Real (192)	Output: Reconstructed speech (same out as V)
Buffer Z	1-63	Long fixed (192)	Output: Coded APC residual
Real Scalar A	0-191	Real	Input: 1st of 2 consecutive scalars: Quant tap, Quant pitch

NUMBER OF OUTPUT SAMPLES: ZBS (=192)
NOTES: This routine use (SA+1) to modify its APS code. SA is first of 2 contiguous scalars (grouped due to limit of 8 function parameters). Special binding is required to handle parameter mixture.
FUNCTION DESCRIPTION:

Generate coded APC residual. See Section 3.12 of the Algorithm Specification (Appendix A of this report) for complete description.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: CALQA(Y,A,U,B,V,C,D)
NAME EXPANSION: Calculate Refined Q and Segment Q's

FCB #: 178
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: CQUS
APS MODULE NAME: CQSS
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Segment Q's
Real Scalar A	0-191	Fixed(in left halfword)	Output: Coded refined frame Q
Buffer U	1-63	Long fixed	Output: Coded normalized segment sums
Real Scalar B	0-191	Real	Input: Frame Q estimate
Buffer V	1-63	Real	Input: Segment sums
Real Scalar C	0-191	Real	Input Frame sum
Real Scalar D	0-191	Real	Input: Maximum segment sum

NUMBER OF OUTPUT SAMPLES: 21
NOTES:

FUNCTION DESCRIPTION:

Generate refined Q, segment q's, and code and quantize as follows:

```
   QR = SB + (5/256)*(SC)          [5/256 = .01953125]
   SA = Coded (QR)
   QRH = Quantized (QR)      } via linear table search
   SEGN(I) = (.998/SD)*V(K) ; I=0,VBS-1
   U(I) = Coded (SEGN(I)) ; I=0,VBS-1
SEGNH(I) = Quantized(SEGNH(I) ; I=9,VBS-1 } via linear table search
   Y(I) = QRH*SEGNH(I) ; I=0,VBS-1
```

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: CORPAR(Y,U)
NAME EXPANSION: Unbitstream, error-correct, and decode the parameters.

FCB #: 123
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: CORPAR

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: an RSOURCE buffer
Buffer U	1-63	Long fixed	Input: an RBITS buffer

NUMBER OF OUTPUT SAMPLES: 17
NOTES: Buffers Y and U must be on Bus 1.

FUNCTION DESCRIPTION:

CORPAR unbitstreams, error-corrects, and decodes the parameters in Buffer U (an RBITS buffer) into Buffer Y (an RSOURCE buffer). The error-correction scheme is taken from APC/SQ05 and 07, which are modifications of the original algorithm.

The formats of the buffers are the same as the buffers for PRORES/PROPAR, which are shown in the source listings. (RSOURCE is Real, whereas TSNK is Long Fixed, but the sample numbers are the same.)

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: COVMX(Y,U,V)
NAME EXPANSION: Computes a (lower triangular) covariance matrix

FCB #: 207
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: COVMSAPU
APS MODULE NAME: COVMSAPS
CSPU MODULE NAME: COVMSSBM

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Matrix Y	1-63	Real (PxP)	Output: Covariance matrix
Buffer U	1-63	Real	Input: Signal
Buffer V	1-63	Real (P+1)	Output: Correlation vector

NUMBER OF OUTPUT SAMPLES: P(P+1)/2 + (P+1)
NOTES: Matrix Y must be configured with MPCLM, not MPCLA. In Release 4, Y may be in the range 1-63. An EXEC patch and a revised MPCLM host support module are required for use under Release 3.5. (CSPI Problem Reports #281 & #1110.)
FUNCTION DESCRIPTION:

This module accepts a real input vector of N samples numbered 0 to N-1, and fills in the elements of a real matrix (lower triangular matrix only), computed according to:

$$y(i,j) = \sum_{k=p}^{N-1} u(k-i)*u(k-j) \qquad \begin{matrix} i=1,2,\ldots P \\ j=i,i+1,\ldots P \end{matrix}$$

The correlation vector is calculated according to:

$$V(i)) = \sum_{k=p}^{N-1} u(k-i)u(k) \qquad i=0,1,\ldots P$$

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: CVMODA(Y,A,U,B,V,W)
NAME EXPANSION: Covariance Matrix Modification

FCB #: 203
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: CVMUS
APS MODULE NAME: CVMSS
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Matrix Y	1-63	Real (4x4)	Output: modified covariance matrix (lower triangular
Scalar A	50-127	Real	Input/Output: Prev/Current frame zeroth autocorrelation term
Buffer U	1-63	Real(5)	Output: Modified correlation vector
Scalar B	50-127	Real	Input: Prev, frame spectrum loop energy reduction
Matrix V	1-63	Real (4x4)	Input: Covariance matrix
Buffer W	1-63	Real (5)	Input: Correlation vector

NUMBER OF OUTPUT SAMPLES: 1 + 5 + 4 * 4 = 22
NOTES: Y and V, U and W may be the same.

FUNCTION DESCRIPTION: This module modifies a 4x4 covariance matrix and accompanying correlation vector. The modification consists of adding to each "the matrix and vector of high pass filtered noise so the prediction coefficient gain will not be excessive. The added components are scaled by the product of 3 numbers, two of which must be supplied as real scalars and the third of which is a constant in the program:

1. The previous frames's zeroth autocorrelation term.
2. The previous frame's spectrum loop energy reduction.
3. A multiplier for the amount of noise (0.1) to be added.

The added matrix and vector, which get multiplied by the three factors described above, are

$$\begin{bmatrix} .375 & 0 & 0 & 0 \\ -.27 & .375 & 0 & 0 \\ .0625 & -.25 & .375 & 0 \\ 0 & .0625 & -.25 & .375 \end{bmatrix} \begin{bmatrix} 0 \\ -.25 \\ .0625 \\ 0 \\ 0 \end{bmatrix}$$

The zeroth element of the input autocorrelation vector 'W' is output as scalar SA, to be used by this module as next frame's "prev. frame zeroth autocorrelation term."

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DAINT
NAME EXPANSION: Simulate D/A Interrupt

FCB #: 127
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: AOMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

The SNAP-callable function calls the AOMINT D/A Interrupt Service module, so it may be used to simulate such an interrupt.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DEALA(U,A,U,B,V,C)
NAME EXPANSION: Deal Received Decoded Data From Single Buffer

FCB #: 182
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: DLU$
APS MODULE NAME: DLS$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (209)	Input: Received decoded data buffer
Real Scalar A	0-191	Real	Output: Decoded frame Q
Buffer U	1-63	Real (10)	Output: Decoded normalized segment sums
Real Scalar B	0-191	Real	Output: Decoded pitch tap
Buffer V	1-63	Real (4)	Output: Decoded reflection coeffs
Real Scalar C	0-191	Real	Output: Decoded pitch lag

NUMBER OF OUTPUT SAMPLES: 17
NOTES:

FUNCTION DESCRIPTION:

On function exit, Y(0-191) is expected to contain 192 received residual samples.

U(0-9) <= Y(192-201)
SA <= Y(202)
SB <= Y(203)
V(0-3) <= Y(204-207)
SC <= Y(208)

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DECRES(Y,U)
NAME EXPANSION: Unbitstream and decode the residual

FCB #: 185
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: DECU$
APS MODULE NAME: DECS$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: an RSOURCE buffer (residual portion only)
Buffer U	1-63	Long fixed	Input: an RBITS buffer

NUMBER OF OUTPUT SAMPLES: YBS
NOTES: Buffer Y must be only the residual portion of the RSOURCE buffer.

FUNCTION DESCRIPTION:

DECRES picks the residual bits out of the RBITS buffer, decodes them using the DVLRL table and taking into account the fact that every 16th residual sample was omitted from the transmission, and outputs them.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DFLON(Y,U,V)
NAME EXPANSION: Digital filter, 0 zeros, N poles

FCB #: 200
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: DFLOU$
APS MODULE NAME: DFLOS$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Filter output (see notes)
Buffer U	1-63	Real	Input: Filter input
Buffer V	1-63	Real	Input: Filter coefficients

NUMBER OF OUTPUT SAMPLES: UBS
NOTES: Buffer Y must be compact, 32-bit floating point. The VBS samples immediately preceding Buffer Y must contain the history of the output signal, i.e., Y(-VBS),..., Y(-1).
FUNCTION DESCRIPTION:

Implements an n-pole filter:

$$y(n) = u(n) = + \sum_{j=0}^{VBS-1} v(j)*y(n-1-j)$$

$$n=0,1,\ldots,UBS-1$$

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: ESTQA(A,U,B,C,D)
NAME EXPANSION: Compute Frame Quantizer Level Estimate

FCB #: 167
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: EQU$
APS MODULE NAME: EQS$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Real Scalar A	0-191	Real	Output: Frame quantizer level estimate
Buffer U	1-63	Real	Input: Quantized RC's
Real Scalar B	0-191	Real	Input: Zeroth autocorrelation value of first residual ÷ frame size
Real Scalar C	0-191	Real	Input: Empirical scale factor
Real Scalar D	0-191	Real	Output: Spectrum loop energy reduction (PROD) ($=V_p$)

NUMBER OF OUTPUT SAMPLES:
NOTES:

FUNCTION DESCRIPTION:

Computes SA as follows:

$$PROD = \prod_{I=0}^{UBS-1} (1-U(I)^2)$$

$$SA = SC * SQRT\ [SB*PROD]$$

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: GTHRA(Y,A,U,B,V,C)
NAME EXPANSION: Gather Transmission Data into Single Buffer

FCB #: 181
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: GTU$
APS MODULE NAME: GTS$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Fixed (209)	Output: Transmission buffer
Real Scalar A	0-191	Fixed (in left half-word)	Input: Coded frame 0
Buffer U	1-63	Fixed (10)	Input: Coded normalized segment sums
Real Scalar B	0-191	Fixed (in left half-word)	Input: Coded pitch tap
Buffer V	1-63	Fixed (4)	Input: Coded reflection coeffs
Real Scalar C	0-191	Fixed (in left half-word)	Input: Coded pitch lag

NUMBER OF OUTPUT SAMPLES: 17
NOTES:

FUNCTION DESCRIPTION:

On function call, Y(0-191) is expected to contain 192 coded residual samples.

Y(192-201) <= U(0-9)
Y(202) <= SA
Y(203) <= SB
Y(204-207) <= V(0-3)
Y(208) <= SC

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: INVPFA(Y,A,U,B)
NAME EXPANSION: Single Tap Inverse Pitch Filter

FCB #: 142
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: IPU$
APS MODULE NAME: IPS$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Filter output (first residual)
Real Scalar A	0-191	Real	Input: Pitch lag (in samples)
Buffer U	1-63	Real (2*YBS)	Input: Filter input: Last frame 2nd APC-spin reconstructed speech, current frame preemphasized speech
Real Scalar B	0-191	Real	Input: Filter coefficient (tap)

NUMBER OF OUTPUT SAMPLES: YBS
NOTES: This routine uses SA to modify its APS code.

FUNCTION DESCRIPTION:

Implements a 1-tap FIR pitch filter:

$$Y(I) = U(I+YBS) - SB*U(I+YBS-SA) \; ; \; I=0, YBS-1$$

Note that Buffer U contains two frames of data, each frame being YBS samples long.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MPGSC(GFLAG,SETCLR)
NAME EXPANSION: G-Flag Set/Clear

FCB #: 106
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: MPGSC

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Literal flag	0-3	Integer	Selects G-flag
Literal SETCLR	0-1	Integer	SETCLR=0=>Set flag SETCLR=1=>Clear flag

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

MPGSC sets or clears one of the four G-flags. This is useful for program timing using an external oscilloscope.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MPIFF(IA,IB,FLID)
NAME EXPANSION: If (IA.NE.0) & (IB.EQ.0) Conditional Function List Execution

FCB #: 105
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: MPIFF$

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Integer Scalar IA	0-127	Integer	Input
Integer Scalar IB	0-127	Integer	Input
Literal FLID	0-63	Integer	Input: Function List ID

NUMBER OF OUTPUT SAMPLES: -
NOTES: Function list 'FLID' must be previously defined.

FUNCTION DESCRIPTION:

Function list 'FLID' is executed if and only if Integer Scalar IA is not equal to zero, and Integer Scalar IB is equal to zero.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MSFBS(Y,U,V,W)
NAME EXPANSION: Symmetric matrix factorization and back-substitution (approximate solution)
FCB #: 202
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: MSFB$APU
APS MODULE NAME: MSFB$APS
CSPU MODULE NAME: MSFB$SBM

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (P)	Output: Pseudo-reflection coefficients
Buffer U	1-63	Real (P+1)	Input: Correlation vector (as output by COVMX).
Matrix V	1-63	Real (PxP)	Input or working: Factored matrix
Matrix W	1-63	Real (PxP)	Input (optional): Covariance matrix. The absence of W is indicated by 0.

NUMBER OF OUTPUT SAMPLES: YBS
NOTES: Matrices V and W must be configured by MPCLM, not MPCLA. In Release 4, Y and W may be in the range 1-63. An Exec patch and a new host support module for MPCLM are required for use under Release 3.5. (CSPI Problem Reports #281 & #1110.)
FUNCTION DESCRIPTION:

If matrix W is given, it is copied to V before the factorization is performed. Then the symmetric matrix V is factored in place into an upper and lower triangular matrix. The correlation vector is then used in a back substitution with the upper triangular matrix to yield an approximate solution for the reflection coefficients. These operations implement the operations specified in the APC/SQ description:

$$S_{ij} = R_{ij} - \sum_{k=1}^{j-1} \frac{S_{ik}S_{jk}}{S_{kk}} \qquad \begin{matrix} i=1,2,\ldots P \\ j=i,i+1,\ldots,P \end{matrix}$$

$$RC_i = [Q_i - \sum_{k=1}^{i=1} RC_k S_{ik}]/S_{ii} \qquad i=1,2,\ldots P$$

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: OVFLWA(Y,A,U)
NAME EXPANSION: Overflow check

FCB #: 199
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: OVU$
APS MODULE NAME: OVS$
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Current frame of output samples, checked for overflow
Real Scalar A	0-191	Real	Input/Output: Last output sample from prev/current frame
Buffer U	1-63	Real	Input: Current frame of putput samples.

NUMBER OF OUTPUT SAMPLES: YBS
NOTES:

FUNCTION DESCRIPTION:

Copies samples from U to Y, replacing any sample U(I) whose absolute value is greater than or equal to 1.0 with the previous output sample Y(I-1). The last output sample in the frame is saved in SA in case of overflow on next frame's U(0).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PCOEF(Y,U)
NAME EXPANSION: Convert Reflection Coefficients to Linear Prediction Coefficients
FCB #: 176
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: VKAU$
APS MODULE NAME: VKASI
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: LP coefficients A(1),A(2),...,A(UBS)
Buffer U	1-63	Real	Input: Reflection Coefficients K(1),K(2),...K(UBS)

NUMBER OF OUTPUT SAMPLES: UBS
NOTES: This function is identical to the VKTOA function in the BBN 16 kb/s MAP-300 system, except for the minus sign in the second equation below; which corresponds to the NSA convention for the signs of the reflection and LP coefficients.
FUNCTION DESCRIPTION:

The LPC coefficients A(k), i = 0,1,...p are computed from the reflection coefficients K(j), j = 1,2,...p, by the recursion:

$$A_m(m) = K(m)$$
$$A_m(L) = A_{m-1}(L) - K(m)A_{m-1}(m-L) \quad \begin{matrix} L = 1,\ldots,m-1 \\ m = 1,\ldots,p \end{matrix}$$

where $A_m(L)$ indicates the value of the L-th coefficient on the m-th iteration

The output buffer does not contain A(0), which is 1.0. The first output value is A(1).

Note: Buffers Y and U may be the same.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PITCHA(Y,A,U,B,C,D)
NAME EXPANSION: Compute AMDF and pitch, code and quantize pitch

FCB #: 134
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: PTU$
APS MODULE NAME: PTS$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (60)	Output: AMDF valves
Real Scalar A	0-191	Real	Output: Quantized pitch lag (in samples)
Buffer U	1-63	Real (285)	Input: Pitch calculation buffer (most recent 285 samples)
Real Scalar B	0-191	Fixed (in left halfword)	Output: Coded pitch lag
Real Scalar C	0-191	Real	Output: Minimum AMDF valve
Real Scalar D	0-191	Real	Output: Maximum AMDF valve

NUMBER OF OUTPUT SAMPLES: 64
NOTES: U Buffer must be compact (USI=2)

FUNCTION DESCRIPTION:

Computes AMDF valves and generates, codes and quantizes pitch. See Section 3.3 of the Algorithm Specification (Appendix A of this report) for complete description.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PROPAR(Y,U)
NAME EXPANSION: Error-protect and bitstream the parameters

FCB #: 122
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: PROPAR

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Long fixed	Output: a TBITS buffer
Buffer U	1-63	Long fixed	Input: a TSNK buffer

NUMBER OF OUTPUT SAMPLES: 59
NOTES: Buffers Y and U must be on Bus 1.

FUNCTION DESCRIPTION:

PROPAR bitstreams the coded parameters from Buffer U (a TSNK buffer) into the appropriate positions in Buffer Y (a TBITS buffer). It also computes and bitstreams 5 parity check bits for an effective (21,16) error-correcting code, using certain significant bits of the coded parameters. The bits protected, and their use in computing the check bits are taken from APC/SQ05, which is a change from the original error-protection algorithm.

The format of the parameter bits in the output buffer is identical to that shown in the APC/SQ documentation. Note that bit 119 (counting from 0) is used for RC4-4, not a second sync bit, as described in some APC/SQ documentation. See the extensive comments in the source listing.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PRRES1(Y,U)
NAME EXPANSION: (Protect and) bitstream the last 1/4 of the residual

FCB #: 121
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: PRRES1

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Long fixed	Output: a TBITS buffer
Buffer U	1-63	Long fixed	Input: a TSNK buffer

NUMBER OF OUTPUT SAMPLES: 45
NOTES: Buffers Y and U must be on Bus 1.

FUNCTION DESCRIPTION:

PRRES1 bitstreams the residual samples from the last 1/4 of Buffer U (a TSNK buffer) into the appropriate positions in Buffer Y (a TBITS buffer). See the PRRES3 function description.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PRRES3(Y,U)
NAME EXPANSION: (Protect and) bitstream the first 3/4 of the residual

FCB #: 120
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: PRRES3

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Long fixed	Output: a TBITS buffer
Buffer U	1-63	Long fixed	Input: a TSNK buffer

NUMBER OF OUTPUT SAMPLES: 135
NOTES: Buffers Y and U must be on Bus 1.

FUNCTION DESCRIPTION:

PRRES3 bitstreams the residual samples from the first 3/4 of Buffer U (a TSNK buffer) into the appropriate positions in Buffer Y (a TBITS buffer). (No error protection is performed; the name is historical.) This function is limited to doing 3/4 of the job in order that its execution time be completely hidden by another function executing in the AP. The remainder of the residual samples are bitstreamed by the PRRES1 function.

The format of the output buffer is similar to that in the published APC/SQ documentation, but the positions of some residual bits are slightly different, due to the omission of every 16th bit out of 192 (rather than every 19th out of 190). See the extensive comments in the source listing for details.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PTSYNA(Y,A,U,B)
NAME EXPANSION: Pitch synthesis filter (single tap)

FCB #: 198
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: PSU$
APS MODULE NAME: PSS$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Filter output: (prev. frame), current frame
Real Scalar A	0-191	Real	Input: Received decoded pitch
Buffer U	1-63	Real	Input: Filter input
Real Scalar B	0-191	Real	Input: Received decoded pitch filter tap.

NUMBER OF OUTPUT SAMPLES: YBS
NOTES: This routine uses SA to modify its APS code. Output Buffer Y is preceded by previous frame data. Buffer Y must be compact (YSI=2)

FUNCTION DESCRIPTION:

Y(I) = U(I) + SB*Y(I-SA)

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: RCALQA(Y,A,U,V)
NAME EXPANSION: Calculate segment and sample Q's

FCB #: 196
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: RQU$
APS MODULE NAME: RQS$
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	*1-63*	*Real (192)*	*Output: Sample Q's*
Real Scalar A	0-191	Real	Input: Decoded received refined frame Q
Buffer U	1-63	Real (10)	Input: Decoded received normalized segment sums
Buffer V	1-63	Real (11)	Input/Output: Segment Q's

NUMBER OF OUTPUT SAMPLES: YBS (=192)
NOTES: This module is very similar to SMPQA

FUNCTION DESCRIPTION:

Calculate Sample Q's by first computing segment Q's, then interpolating as in module SMPQA.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: RCCQA(Y,U,V)
NAME EXPANSION: Reflection Coefficient Coding and Quantization

FCB #: 150
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: RCU$
APS MODULE NAME: RCS$
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Quantized RC's
Buffer U	1-63	Long Fixed	Output: Coded RC's
Buffer V	1-63	Real	Input: Unquantized RC's

NUMBER OF OUTPUT SAMPLES: YBS
NOTES:

FUNCTION DESCRIPTION:

This module codes (to 4 bits) and quantizes the input reflection coefficients via linear comparisons against entries in threshold value table.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: REMDCA (Y,A,U,B)
NAME EXPANSION: Remove Long-Term DC Bias

FCB #: 133
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: RMU$
APS MODULE NAME: RMS$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: DC-removed input speech
Real Scalar A	0-191	Real	Input/Output: Running DC sum
Buffer U	1-63	Real	Input: Input speech (range [-1,1])
Real Scalar B	0-191	Real	Input/Output: Running DC Value

NUMBER OF OUTPUT SAMPLES: UBS + 2 Scalars
NOTES: This function is similar, but not identical, to the first part of the LPC-10 routine "DCRZCL". This function uses the first sample in each frame for DC estimation; LPC-10's uses the last sample in each frame.
FUNCTION DESCRIPTION:

```
SA' = SA + U(0) - SB
TEMP = SA'/(2**10)
IF ( TEMP-SB|<2**-11) SB' = SB
ELSE
[IF (TEMP>SB) SB' = SB + (2**-11)
[IF (TEMP<SB) SB' = SB - (2**-11)
Y(I) = U(I)-SB' ; I = 0, UBS-1
```

Note: For 11-bit samples (plus sign) made fractional by dividing by 2**11, +1/(2**11)) is the smallest sample increment, equivalent to ±1 for integer samples.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: RMINT
NAME EXPANSION: Simulate Receiver Modem Interrupt

FCB #: 126
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: RMODEMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

This SNAP-callable function calls the RMODEMINT Receiver Modem Interrupt Service module, so it may be used to simulate such an interrupt.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: SHISTA(S,T,U,V,W,X)
NAME EXPANSION: Set up synthesis frame histories

FCB #: 190
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: SHU$
APS MODULE NAME: SHS$
CSPU MODULE NAME: SBM$SH

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer S	1-63	Real	Output: First segment Q
Buffer T	1-63	Real	Input: Last segment Q
Buffer U	1-63	Real	Output: Pitch filter history: last, current frames
Buffer V	1-63	Real	Input: Pitch filter history: current frame
Buffer W	1-63	Real	Output: Synth. filter history: last 4 samples of previous frame
Buffer X	1-63	Real	Input: Synth filter history: last 4 samples of current frame

NUMBER OF OUTPUT SAMPLES: VBS + XBS
NOTES: Uses special binding for programming convenience.

FUNCTION DESCRIPTION:

S(0) - S(TBS-1) <= T(0) - T(TBS-1)

U(0) - U(VBS-1) <= V(0) - V(VBS-1)

W(0) - W(XBS-1) <= X(0) - X(XBS-1)

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: SMPQA(Y,U)
NAME EXPANSION: Calculate Sample Q's

FCB #: 179
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: SQU$
APS MODULE NAME: SQS$
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (192)	Output: Sample Q's
Buffer U	1-63	Real (11)	Input: Segment Q's (with history)

NUMBER OF OUTPUT SAMPLES: YBS
NOTES:

FUNCTION DESCRIPTION:

Interpolates between segment Q's to get sample by sample Q's.

```
D(0) = (U(1) - U(0))/9     [19/2 APPROX = 9]
Y(0) = U(0)
Y(J) = Y(J-1) + D(0) ; J=1,...,8
FOR I=1,...,9
   D(I) = (U(I+1) - U(I))/19
   Y(9 + (I-1)*19) = U(I)
   Y(J) = Y(J-1) + D(I) ; J=10+(I-1)*19,...,27+(I-1)*19
END FOR
   (D(10) = 0)
   Y(180) = U(10)
   Y(J) = Y(J-1) ; J=181,...,191)
```

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: TAPA(Y,A,B,C,D)
NAME EXPANSION: Compute Single Pitch Predictor Coefficient ("Tap"), Code and Quantize
FCB #: 135
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: TPU$
APS MODULE NAME: TPU$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Input: Current frame preemphasized speech, assumed preceded by last frame 2nd APC-spin reconstructed speech
Real Scalar A	0-191	Fixed (in left halfword)	Input/Output: Coded pitch lag
Real Scalar B	0-191	Real	Input: Quantized pitch lag (in samples)
Real Scalar C	0-191	Fixed (in left halfword)	Output: coded pitch tap
Real Scalar D	0-191	Real	Output: Quantized pitch tap

NUMBER OF OUTPUT SAMPLES: 3
NOTES: This routine uses SB to modify its APS code. Buffer Y must be compact (YSI=2).

FUNCTION DESCRIPTION:

Tap is computed as follows:

```
BEGIN
    SUM2 =   YBS-1
               Σ      [Y(I)*Y(I-SB)]     (where array indices start at 0)
           I=YBS/2
    IF (SUM2 .LE. 0)
       TAP=0
       EXIT
    SUM1 =  YBS-1
              Σ     [Y(I-SB)*Y(I-SB)]
          I=YBS/2
    IF (SUM1 .EQ. 0)
       TAP=0
       EXIT
    TAP = SUM2/SUM1
EXIT
```

This value for tap is then coded and quantized by comparing it linearly to a quantization threshold table (A tap of zero is coded and quantized as zero, and in addition, coded pitch is set to zero.) Coded tap and quantized tap are generated from internal coded value and quantization value tables accessed in parallel with the threshold table. Hardware AP flags WI and FWI are used throughout for APU/APS synchronization.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: TMINT
NAME EXPANSION: Simulate Transmitter Modem Interrupt

FCB #: 125
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: TMODEMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

This SNAP-callable function calls the TMODEMINT Transmitter Modem Interrupt Service module, so it may be used to simulate such an interrupt.

APPENDIX C
BBN-WRITTEN MAP-300 FUNCTIONS FOR LPC-10

This Appendix contains a complete user-level description for each MAP-300 SNAP-II function written by BBN and used in the LPC-10 speech coder implementation. The descriptions are ordered alphabetically by function name.

Functions that were used in the speech coder implementation and are not described here were supplied by CSPI as part of the SNAP-II software System (See Sections 2.5.1.1 and 2.5.2.1).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: ADINT
NAME EXPANSION: Simulate A/D interrupt

FCB #: 124
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: ADAMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

This SNAP callable function calls the ADAMINT A/D Interrupt Service module in exactly the same manner as it is used to respond to A/D interrupts, so it may be used to simulate such an interrupt.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: AHISTA (S,T,U,V,W,X,Y,Z)
NAME EXPANSION: Set up frame histories.

FCB #: 132
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: AHU$
APS MODULE NAME: AHS$
CSPU MODULE NAME: SBM$AH

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer S	1-63	Real	Output: Current data and history
Buffer T	1-63	Real	Input: History data
Buffer U	1-63	Real	Output: Current data and history
Buffer V	1-63	Real	Input: History data
Buffer W	1-63	Real	Output: Current data and history
Buffer X	1-63	Real	Input: History data
Buffer Y	1-63	Real	Output: Current data and history
Buffer Z	1-63	Real	Input: History data

NUMBER OF OUTPUT SAMPLES: TBS + VBS + XBS + ZBS
NOTES: Requires special CSPU binding

FUNCTION DESCRIPTION:

S(0) - S(TBS-1) <= T(0) - T(TBS-1)
U(0) - U(VBS-1) <= V(0) - V(VBS-1)
W(0) - W(XBS-1) <= X(0) - X(XBS-1)
Y(0) - Y(ZBS-1) <= Z(0) - Z(ZBS-1)

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: APHRML(Y,A,U,B,V,C,D)
NAME EXPANSION: Compute a semi-pitch synchronous interval for LPC Analysis; compute an RMS Energy and Code the Energy for transmission
FCB #: 178
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: ARU$
APS MODULE NAME: ARS$
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: LPC analysis interval samples
Real Scalar A	0-191	Real (4)	Input: 1st of 4 consec. scalars defining frame lengths
Buffer U	1-63	Real (286)	Input: DC removed input speech samples (current frames and history)
Real Scalar B	0-91	Real (2)	Input/output: 1st of 2 consec. scalars defining pitch-synchronous interval locations
Buffer V	1-63	Real (9)	Output: Energy parameter buffer
Real Scalar C	0-191	Real	Input: Voicing decision-current frame
Real Scalar D	0-191	Real	Input: Pitch lag - current frame

NUMBER OF OUTPUT SAMPLES: YBS+SB1+3 components of V

NOTES: This routine modifies the APS code

FUNCTION DESCRIPTION:

The location of a pitch synchronous interval (length=130 samples) is determined for LPC analysis. The samples in the interval are preemphasized and the DC is removed from them. A mean-square energy is also computed from a subset of samples within the analysis interval and this energy is coded for transmission.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: COVMX(Y,U,V)
NAME EXPANSION: *Computes a (lower triangular) covariance matrix*

FCB #: 198
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: COVM$APU
APS MODULE NAME: COVM$APS
CSPU MODULE NAME: COVM$SBM

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Matrix Y	1-63	Real (PxP)	OUtput: Covariance matrix
Buffer U	1-63	Real	Input: Signal
Buffer V	1-63	Real (P+1)	Output: Correlation vector

NUMBER OF OUTPUT SAMPLES: *P(P+1)/2 + (P+1)*
NOTES: Matrix Y must be configured with MPCLM, not MPCLA. In Release 4, Y may be in tne range 1-63. An EXEC patch and a revised MPCLM host support module are required for use under Release 3.5. (CSPI Problem Reports #281 & #1110.)
FUNCTION DESCRIPTION:

This module accepts a real input vector of N samples numbered 0 to N-1, and fills in the elements of a real matrix (lower triangular matrix only), computed according to:

$$y(i,j) = \sum_{k=p}^{N-1} u(k-i)*u(k-j) \qquad \begin{matrix} i=1,2,\ldots P \\ j=i,i+1,\ldots P \end{matrix}$$

The correlation vector is calculated according to:

$$V(i)) = \sum_{k=p}^{N-1} u(k-i)u(k) \qquad i=0,1,\ldots P$$

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DAINT
NAME EXPANSION: Simulate D/A Interrupt

FCB #: 127
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: AOMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

The SNAP-callable function calls the AOMINT D/A Interrupt Service module, so it may be used to simulate such an interrupt.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DCRZCL(Y,A,U,B,C,D)
NAME EXPANSION: Peak signal scan, DC removal, zero-crossing count

FCB #: 133
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: DCU$
APS MODULE NAME: DCS$
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (286)	Input/output: Long term DC removed speech samples - current frame and history
Real Scalar A	0-191	Real	Input/output: Running DC Sum
Buffer U	1-63	Real	Input: Input samples - current frame
Real Scalar B		Real	Input/output: DC value
Real Scalar C	0-191	Fixed (3) in lefthword)	Output: Zero crossing counts 1st half frame (SC1), 2nd half frame (SC2), and the sum=SC3=SC1+SC2
Real Scalar D	0-191	Fixed (in lefthword)	Input: Dither factor

NUMBER OF OUTPUT SAMPLES: UBS+5

NOTES:

FUNCTION DESCRIPTION:

Update a running DC sum and DC value and remove the upated DC value from the input samples. Performs a running maximum-absolute value scan of the input samples and updates the maximum which is maintained in a hardwired integer scalar TADPK. Also this routine computes the zero crossing counts for the first and second half of the pitch analysis interval.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DYNPRL (Y,A,U,B,V,C,D)
NAME EXPANSION: Dynamic Programming Routine for Pitch Extraction and Coding

FCB #: 179
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: DPLU$
APS MODULE NAME: DPLS$
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (60)	Input/Output: "Winner" function
Real Scalar A	0-191	Real (2)	Input/Output: Confidence factor SA1 Output:Scaled Confidence factor SA2
Buffer U	1-63	Real (60)	Input: AMDF function
Real Scalar B	0-191	Real (3)	Input: 3 Voicing decisions
Buffer V	1-63	Real (180)	Input/Output: 3 Consecutive Pointer Vectors V1, V2, V3
Real Scalar C	0-191	Real	Input: Minimum of AMDF function
Real Scalar D	0-191	Real (6) (SD3-fixed in L half word)	Output: 6 Consecutive Scalars- Winner function (min and max) pitch code, indexes (tx and current) and pitch lag (current)

NUMBER OF OUTPUT SAMPLES: YBS + 3 * VBS + 8
NOTES: I/O are integrated into a single instruction stream. Buffers Y,U, and V must be Real, compact and of equal and fixed length

FUNCTION DESCRIPTION:

The voicing state of the transmitted frame is determined. For voiced frames, the confidence factor (SA1) is updated with the minimum of the AMDF function and then scaled by 1/16. The "Winner" function (Y) is updated and the current frame Pointer Vector (V1) is evaluated. The "Winner" function is then updated with the current AMDF function (U) and normalized for zero minimum for the next call to DYNPRL. A trace back through the pointer vectors is made to determine the pitch index for the transmitted frame. The transmitted pitch is coded and output.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: ECORDL (Y,A,U)
NAME EXPANSION: Error correction, smoothing, and decoding

FCB #: 109
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: ECORDL

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Two 15-fullword buffers containing synthesis parameters
Integer Scalar A	0-127	Integer	Bitstream buffer (RBITS) flag
Buffer U	1-63	Long fixed	Input: Bitstream (RBITS) buffer

NUMBER OF OUTPUT SAMPLES: 30
NOTES: Both buffers must be compact and on Bus 1. In addition to the arguments, there are 3 integer scalars (97-99) and 8 15-halfword buffers that are used as input/output only by ECORDL.

FUNCTION DESCRIPTION:
ECORDL unbitstreams the serial bit of data of the received frame and then clears the bitstream buffer flag to signify that the buffer is empty.

The received coded parameters are error-corrected and smoothed to combat the effect of channel errors and then decoded into the first half of the output buffer Y. In addition, a set of interpolated reflection coefficients are output into the second half of Y. The format of Buffer U is given in Table 4 of the text. The format of Buffer Y is:

```
Y(0):        Pitch (integer in left half)
Y(2):        RMS code (6-bit integer in left half)
Y(3)-Y(12):  Reflection coefficients
Y(13):       1st half frame voicing (0=unvoiced, 1=voiced)
Y(14):       2nd half frame voicing (0=unvoiced, 1=voiced)
Y(18)-Y(27): Interpolated reflection coefficients
```

The error-correction and parameter-smoothing heuristics are described briefly in Section 2.3.2.1 and more completely in [12, pp. 126-144].

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: EPOCHL (Y,U,V)
NAME EXPANSION: Compute epoch tables for pitch-synchronous synthesis

FCB #: 110
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: EPOCHL

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Up to 11 13-word epoch tables
Buffer U	1-63	Long fixed	Output: Excitation
Buffer V	1-63	Real	Input: Triple buffer of frame parameters

NUMBER OF OUTPUT SAMPLES: Variable
NOTES: All buffers must be compact and on Bus 1. Buffer Y must be at least 144 words long. The 156 halfwords preceding Buffer U are used for "end of previous frame" excitation.
FUNCTION DESCRIPTION:
Based on the frame parameter data in Buffer V, EPOCHL generates an excitation signal (Buffer U) and synthesis parameters in pitch-synchronous "epochs" (Buffer Y). The details of the epoch generation depend on the voicing of the previous half frame and the two current half frames [12, pp. 145-159].

The first 30 words of Buffer V are the same as Buffer Y of ECORDL, except that the reflection coefficients have been converted to predictor coefficients. The last 15 words of Buffer V are the first 15 words from Buffer V of the previous frame.

The format of each 13-word epoch table in Buffer Y is:

Y(0):	Epoch duration (integer in left half); a negative value terminates the epoch table.
Y(1):	RMS value
Y(2)-Y(11):	Predictor coefficients
Y(12):	Offset in the excitation input and synthesis output buffers for the start of the epoch (sign and magnitude integer in left half).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: *EPROL (Y,U)*
NAME EXPANSION: Error protect and bitstream the coded frame parameters

FCB #: 122
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: EPROL

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Long fixed	Output: serial transmission data
Buffer U	1-63	Long fixed	Input: coded parameters

NUMBER OF OUTPUT SAMPLES: 54
NOTES: Both buffers must be compact and on Bus 1

FUNCTION DESCRIPTION:
EPROL error-protects (unvoiced and transition frames only) and bitstreams the coded frame parameters in Buffer U, producing serial-transmission data in Buffer Y.

The format of Buffer U is:

U(0):	Pitch/voicing (7 bits)
U(1):	RMS (5 bits)
U(2-5):	K1-K4 (5 bits)
U(6-9):	K5-K8 (4 bits)
U(10):	K9 (3 bits)
U(11):	K10 (2 bits)

The format of Buffer Y is shown in Table 4 of the report.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: INVFTL(Y,U,V,W)
NAME EXPANSION: Second Order Inverse Filter

FCB #: 167
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: IFLU$
APS MODULE NAME: IFLS$
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Inverse filtered speech samples
Buffer U	1-63	Real (7)	Output: Inverse filter parameter: (3) autocorrelation coefficients (2) reflection coefficients (2) predictor coefficients
Buffer V	1-63	Real	Input: Lowpass filtered speech samples (current frame-even samples
Buffer W	1-63	Real	Input: Lowpass filtered speech samples (current frame-odd samples

NUMBER OF OUTPUT SAMPLES: UBS+UBS
NOTES:

FUNCTION DESCRIPTION:

Three autocorrelation coefficients with lags of zero, four and eight are computed from the lowpass filtered input speech buffers V and W. (Note: if the zero lag autocorrelation is equal to zero, the lowpass filtered speech buffers V and W are copied to the output buffer Y and filtering is aborted.) Two reflection coefficients are computed from the three autocorrelation coefficients and clipped to ± 1. Two predictor coefficients are then computed and the input lowpass filtered speech samples are inverse filtered in two passes as

$$Y(j) = V(i)-U(6)*V(i-2)-U(7)*V(i-4), \begin{matrix} i=\emptyset,1,2,\ldots,VBS-1 \\ j=0,2,4,\ldots,YBS-2 \end{matrix}$$

$$Y(j) = W(i)-U(6)*W(i-2)-U(7)*W(i-4), \begin{matrix} i=\emptyset,1,2,\ldots,WBS-1 \\ j=1,3,5,\ldots,YBS-1 \end{matrix}$$

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: LPBUTL(Y,U,V,W)
NAME EXPANSION: *4th Order Low-Pass Butterworth Filter*

FCB #: 150
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: LPLU$
APS MODULE NAME: LPLS$
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Input/output: Lowpass filtered speech samples (current frame)
Buffer U	1-63	Real	Input: Long term DC removed speech samples (current frame)
Buffer V	1-63	Real (4)	Input: Filter coefficients (4)
Buffer W	1-63	Real	Output: Lowpass filtered speech samples, scaled by 1/16 (current frame)

NUMBER OF OUTPUT SAMPLES: YBS+WBS+4
NOTES: Y must be compact, 32-bit floating point. The VBS samples immediately preceding Buffer Y must contain the previous history of the unscaled output signal, I.E., Y(-VBS),..,Y(-2), Y(-1).
FUNCTION DESCRIPTION:

The input signal, U, is lowpass filtered by a 4th order Butterworth filter. The filtered signal is output into buffer Y,

$$Y(i) = .3069*U(i) + \sum_{k=0}^{3} V(k)*Y(i-1-k),\ i=\emptyset,YBS-1$$

The filtered signal Y is scaled down by a factor 1/16: the scaled signal is stored in buffer W

$$W(i) = Y(i)*1/16,\ i=\emptyset,\ WBS-1$$

The filter history is also updated and stored in the 4 memory locations immediately preceding the buffer Y,

$$Y(-4) = Y(YBS-4)$$
$$Y(-3) = Y(YBS-3)$$
$$Y(-2) = Y(YBS-2)$$
$$Y(-1) = Y(YBS-1)$$

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MPGSC(GFLAG,SETCLR)
NAME EXPANSION: G-Flag Set/Clear

FCB #: 106
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: MPGSC

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Literal flag	0-3	Integer	Selects G-flag
Literal SETCLR	0-1	Integer	SETCLR=0=>Set flag SETCLR=1=>Clear flag

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

MPGSC sets or clears one of the four G-flags. This is useful for program timing using an external oscilloscope.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MPIFF(IA,IB,FLID)
NAME EXPANSION: If (IA.NE.0) & (IB.EQ.0) Conditional Function List Execution

FCB #: 105
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: MPIFF$

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Integer Scalar IA	0-127	Integer	Input
Integer Scalar IB	0-127	Integer	Input
Literal FLID	0-63	Integer	Input: Function List ID

NUMBER OF OUTPUT SAMPLES: -
NOTES: Function list 'FLID' must be previously defined.

FUNCTION DESCRIPTION:

Function list 'FLID' is executed if and only if Integer Scalar IA is not equal to zero, and Integer Scalar IB is equal to zero.

AD-A116 902 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2
REALTIME IMPLEMENTATION OF THE APC/SQ AND LPC-10 SPEECH CODING --ETC(U)
JUN 82 J J WOLF, K D FIELD, W H RUSSELL DCA100-80-C-0034
UNCLASSIFIED BBN-4855 NL

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MSFBS(Y,U,V,W)
NAME EXPANSION: Symmetric matrix factorization and back-substitution (approximate solution)
FCB #: 199
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: MSFB$APU
APS MODULE NAME: MSFB$APS
CSPU MODULE NAME: MSFB$SBM

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (P)	Output: Pseudo-reflection coefficients
Buffer U	1-63	Real (P+1)	Input: Correlation vector (as output by COVMX).
Matrix V	1-63	Real (PxP)	Input or working: Factored matrix
Matrix W	1-63	Real (PxP)	Input (optional): Covariance matrix. The absence of W is indicated by 0.

NUMBER OF OUTPUT SAMPLES: YBS
NOTES: Matrices V and W must be configured by MPCLM, not MPCLA. In Release 4, Y and W may be in the range 1-63. An Exec patch and a new host support module for MPCLM are required for use under Release 3.5. (CSPI Problem Reports #281 & #1110.)
FUNCTION DESCRIPTION:

If matrix W is given, it is copied to V before the factorization is performed. Then the symmetric matrix V is factored in place into an upper and lower triangular matrix. The correlation vector is then used in a back substitution with the upper triangular matrix to yield an approximate solution for the reflection coefficients. These operations implement the operations specified in the APC/SQ description:

$$S_{ij} = R_{ij} - \sum_{k=1}^{j-1} \frac{S_{ik}S_{jk}}{S_{kk}} \qquad \begin{matrix} i=1,2,\ldots P \\ j=i,i+1,\ldots,P \end{matrix}$$

$$RC_i = [Q_i - \sum_{k=1}^{i=1} RC_k S_{ik}]/S_{ii} \qquad i=1,2,\ldots P$$

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PCOEF2(Y,U,V,W)
NAME EXPANSION: Convert ReflectionCoefficients to Linear Prediction Coefficients, 2 sets
FCB #: 196
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: VKAU$
APS MODULE NAME: VKASI
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: LP coefficients A(1),A(2),...,A(UBS)
Buffer U	1-63	Real	Input: Reflection coefficients K(1),K(2),...K(UBS)
Buffer V	1-63	Real	Output: LP coefficients A(1),A(2),...,A(UBS)
Buffer W	1-63	Real	Input: Reflection coefficients K(1),K(2),...K(UBS)

NUMBER OF OUTPUT SAMPLES: UBS
NOTES: This function is identical to the PCOEF function in the APC/SQ system, except that it does two conversions, from buffer U to buffer Y and from buffer W to buffer V
FUNCTION DESCRIPTION:

The LPC coefficients A(k), i=0,1,...p are computed from the reflection coefficients K(j), j= 1,2,...p, by the recursion:

$$A_m(m) = K(m)$$
$$A_m(L) = A_{m-1}(L) - K(m)A_{m-1}(m-L) \quad \begin{matrix} L = 1,\ldots,m-1 \\ m = 1,\ldots,p \end{matrix}$$

where $A_m(L)$ indicates the value of the L-th coefficient on the m-th iteration. The output buffer does not contain A(0), which is 1.0. The first output value is A(1).
Note: Buffers Y and U may be the same, as may V and W.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PITCHA(Y,A,U,B,C,D)
NAME EXPANSION: Compute AMDF and pitch, code and quantize pitch

FCB #: 177
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: PTU$
APS MODULE NAME: PTS$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (60)	Output: AMDF valves
Real Scalar A	0-191	Real	Output: Quantized pitch lag (in samples)
Buffer U	1-63	Real (285)	Input: Pitch calculation buffer (most recent 285 samples)
Real Scalar B	0-191	Fixed (in left halfword)	Output: Coded pitch lag
Real Scalar C	0-191	Real	Output: Minimum AMDF valve
Real Scalar D	0-191	Real	Output: Maximum AMDF valve

NUMBER OF OUTPUT SAMPLES: 64
NOTES: U Buffer must be compact (USI=2)

FUNCTION DESCRIPTION:

Computes AMDF valves and generates, codes and quantizes pitch. See Section 3.3 of the Algorithm Specification (Appendix A of this report) for complete description.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: RCCGTL(Y,A,U,V)
NAME EXPANSION: Code Reflection Coefficients and gather all transmitted data into a single buffer
FCB #: 182
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: RCU$
APS MODULE NAME: RCS$
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Fixed	Output: Transmission buffer
Real Scalar A	0-191	Fixed (in lefthword)	Input: Coded pitch
Buffer U	1-63	Real (10)	Input: Reflection coefficients (three frames-current, past and transmitted)
Buffer V	1-63	Fixed (1) (in lefthword)	Input: coded gain - (Third entry in buffer of length 9)

NUMBER OF OUTPUT SAMPLES: YBS
NOTES: Used only part of input buffer U and V

FUNCTION DESCRIPTION: Code reflection coefficients and gather transmitted parameters into a single buffer Y for transmission. The order of the parameters in buffer Y is pitch, gain and the ten reflection coefficients.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: RHISTB (Y,U,V,W)
NAME EXPANSION: Set up frame histories.

ICB #: 135
ARRAY OR NON-ARRAY FUNCTION: Array

AFU MODULE NAME: RHBU$
APS MODULE NAME: RHBS$
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Current data and histo:
Buffer U	1-63	Real	Input: History data
Buffer V	1-63	Real	Output: Current data and histo:
Buffer W	1-63	Real	Input: History data

NUMBER OF OUTPUT SAMPLES: UBS + WBS
NOTES:

FUNCTION DESCRIPTION:

Y(0) - Y(UBS-1)<= U(0) - U(UBS-1)

V(0) - V(WBS-1)<= W(0) - W(WBS-1)

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: RHISTC (Y,U)
NAME EXPANSION: Set up frame histories.

FCB #: 142
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: RHCU$
APS MODULE NAME: RHCS$
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Current data and histor
Buffer U	1-63	Real	Input: History data

NUMBER OF OUTPUT SAMPLES: UBS
NOTES:

FUNCTION DESCRIPTION:

Y(0) - Y(UBS-1)<= U(0) - U(UBS-1)

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: RMINT
NAME EXPANSION: Simulate Receiver Modem Interrupt

FCB #: 126
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: RMODEMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

This SNAP-callable function calls the RMODEMINT Receiver Modem Interrupt Service module, so it may be used to simulate such an interrupt.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: SHISTA(S,T,U,V,W,X)
NAME EXPANSION: Set up frame histories

FCB #: 134
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: SHU$
APS MODULE NAME: SHS$
CSPU MODULE NAME: SBM$SH

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer S	1-63	Real	Output: Current data and history
Buffer T	1-63	Real	Input: History data
Buffer U	1-63	Real	Output: Current data and history
Buffer V	1-63	Real	Input: History data
Buffer W	1-63	Real	Output: Current data and history
Buffer X	1-63	Real	Input: History data

NUMBER OF OUTPUT SAMPLES: TBS+VBS + XBS
NOTES: Requires special CSPU binding

FUNCTION DESCRIPTION:

S(0) - S(TBS-1)<= T(0) - T(TBS-1)
U(0) - U(VBS-1)<= V(0) - V(VBS-1)
W(0) - W(XBS-1)<= X(0) - X(XBS-1)

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: SSMABL (U,A,V,B)
NAME EXPANSION: Compute 1st and 2nd half frame energy measures

FCB #: 176
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: SSMU$
APS MODULE NAME: SSMS$
CSPU MODULE NAME:

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer u	1-63	Real	Input: Lowpass filtered speech samples, 1st half frame
Real Scalar A	0-191	Fixed (in lefthword)	Output: Energy measure, 1st half frame
Buffer V	1-63	Real	Input: Lowpass filtered speech samples, 2nd half frame
Real Scalar B	0-191	Fixed (in lefthword)	Output: Energy measure 2nd half frame

NUMBER OF OUTPUT SAMPLES: 2
NOTES:

FUNCTION DESCRIPTION:

Energy measures are computed as the sum of the absolute value of the lowpass filtered speech samples as

```
A = SUM [ABS(U(i))], i=ø, UBS-1
B = SUM [ABS(V(i))], i=ø, VBS-1
```

the energy measures (A&B) are then scaled by 2**11 and converted to an integer

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: SYNSEL(RBF,SLCTR,FLG0,FLG1,FLG2,FLID0,FLID1,FLID2)
NAME EXPANSION: Synthesis function list selection

FCB #: 107
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: SYNSEL$

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Integer Scalar RBF	0-127	Integer	Input: RBITS buffer flag
Integer Scalar SLCTR	0-127	Integer	Input/Output: Selector (0,1,2
Integer Scalar FLG0	0-127	Integer	Input: ROS buffer flag
Integer Scalar FLG1	0-127	Integer	Input: ROS buffer flag
Integer Scalar FLG2	0-127	Integer	Input: ROS buffer flag
Literal FLID0	0-63	Integer	Input: Synthesis function list ID
Literal FLID1	0-63	Integer	Input: Synthesis function list ID
Literal FLID2	0-63	Integer	Input: Synthesis function list ID

NUMBER OF OUTPUT SAMPLES: --
NOTES: Function lists 'FLID0','FLID1', and 'FLID2' must have been defined.

FUNCTION DESCRIPTION:

Selects the next synthesis function list (in the order (0,1,2,0,1,...) based on the value of SLCTR) and executes it if and only if the integer scalar RBF is not equal to 0 and the proper two ROS buffer flags are both zero; if the function list is executed, steps the value of SLCTR to the next value in the sequence, as illustrated by the Fortran-77 fragment below:

```
      IF(RBF.NE.0) THEN
        IF(SLCTR.EQ.0) THEN
          IF(FLG0.EQ.0.AND.FLG1.EQ.0) THEN
            SLCTR=1
            CALL MPXFL(FLID0)
          ENDIF
        ELSE IF(SLCTR.EQ.1) THEN
          IF(FLG1.EQ.0.AND.FLG0.EQ.0) THEN
            SLCTR=0
            CALL MPXFL(FLID1)
          ENDIF
        ELSE IF(SLCTR.EQ.2) THEN
          IF(FLG2.EQ.0.AND.FLG0.EQ.0) THEN
            SLCTR=0
            CALL MPXFL(FLID2)
          ENDIF
        ENDIF
      ENDIF
```

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: SYNTHL (Y,A,U,B,V,C,W)
NAME EXPANSION: Pitch-synchronous synthesis, including gain correction and de-emphasis
FCB #: 202
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: SYNU$
APS MODULE NAME: SYNS$
CSPU MODULE NAME: SBM$SYN

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real, short	Output: Output speech
Scalar A	50-127	Real	Input/Output: De-emphasis history
Buffer U	1-63	Real	Input: Up to 11 13-word epoch table
Scalar B	50-127	Real	Input/Output: Previous epoch RMS
Buffer V	1-63	Real	Input/Output: Scratch
Integer Scalar C	0-127	Integer	Output: Buffer status flag
Buffer W	1-63	Long fixed	Input: Excitation signal

NUMBER OF OUTPUT SAMPLES: Variable
NOTES: All four buffers must be compact. Buffers Y, U, and W must be on Bus 1.

FUNCTION DESCRIPTION: Performs pitch-synchronous synthesis, followed by gain correction and fixed de-emphasis.

Buffer formats:

Y and W: 180 halfwords long, with previous 156 halfwords also available for use. These "negative index" data are for filling out part of the previous output buffer that was not filled by SYNTHL on the previous frame.

V: 156 words long, with previous 10 words also available (for filter history).

U: Up to 11 13-word epoch tables. The format of each epoch table is:

- u(0): Epoch duration (integer in left half); a negative value terminates the epoch table.
- u(1): RMS value.
- u(2)-u(11): Predictor coefficients.
- u(12): Offset in Buffers Y and W of the start of the epoch (sign and magnitude integer in left half), $-156 \leq u(12) \leq 159$.

For each epoch specified in Buffer U:

1. If the epoch duration u(0) < 0, then set Integer Scalar C to 1 to signify the old synthesis output buffer is full, then exit.

2. Find epoch rescaling factor.
   if u(1) < 1.0 then u(1)=1.0
   RMSCAL = SB/u(1)

output SB = u(1)
if RMSCAL > 8.0, then RMSCAL = 8.0

3. Rescale 10-pole filter history

$V(i) = V(i)*RMSCAL \qquad i=-10,-9,\ldots,-1$

4. Do 10-pole synthesis

$$V(i) = W(i + u(12)) + \sum_{k=0}^{9} V(i - k - 1) * u(2 + k) \qquad i=0,1,\ldots,u(0)-1$$

5. Update 10-pole filter history

$V(i) = V(u(0) + i) \qquad i=-10,-9,\ldots,-1$

6. Compute gain scaling factor

$$VRMS = \sum_{i=0}^{u(0)-1} V(I)$$

$$XRMS = \sqrt{VRMS/u(0)}$$

$$S = \frac{u(1)}{2^{11} * XRMS}$$

7. Rescale and de-emphasize the output

```
HIST = SA
for i=0,1,...,u(0)-1:
    TEMP = S * V(i) + 0.75 * HIST
    if TEMP > 1.0, then TEMP = HIST
    Y(i+u(12)) = TEMP
    HIST = TEMP
```

8. Output de-emphasis history

SA = HIST.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: TMINT
NAME EXPANSION: Simulate Transmitter Modem Interrupt

FCB #: 125
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: TMODEMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

This SNAP-callable function calls the TMODEMINT Transmitter Modem Interrupt Service module, so it may be used to simulate such an interrupt.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: VNEWL (A,B,C,D)
NAME EXPANSION: Compute new voicing decisions and update old ones

FCB #: 108
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: VNEWL

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Real Scalar A	50-127	Real	Input: first of 5 consecutive scalars, each of which contains an integer in its LH: TNZ1, TNZ2, TNZT, TEM1, TEM2.
Real Scalar B	50-127	Real	Input: First of 2 consecutive scalars: TAMN, TAMZ
Real Scalar C	50-127	Real	Output: Integer in LH: TDTH
Real Scalar D	50-127	Real	Output: First of 3 consecutive scalars: TVD1, TVD2, TVDS

NUMBER OF OUTPUT SAMPLES: 4
NOTES: In addition, there are 2 blocks of integer scalars (89-96, 98-104) that are used as input/output only by VNEWL.

FUNCTION DESCRIPTION:
VNEWL executes voicing decision logic for the two halves of the current frame, and it refines the voicing decisions for the two halves of the frame two in the past, which is the one about to be transmitted.

For a more complete description of the input and output parameters, see Sec. 2.3.1.5 and the LPC10M.MSO program source file. The voicing logic is rather involved, and it is well documented in [12, pp. 69-89].